VŠB – TECHNICKÁ UNIVERZITA OSTRAVA

FAKULTA STROJNÍ

KATEDRA AUTOMATIZIČNÍ TECHNIKY a ŘÍZENÍ

Využití .NET webových služeb v prostředí automatizace

Vedoucí diplomové práce: Ing. Marek Babiuch, Ph.D.

Diplomant: Bc. Martin Hnik

Ostrava: 22. 5. 2009

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Využití .NET webových služeb v prostředí automatizace

Using .NET web Services at Area of Automation

Student:	**Bc. Martin HNIK**
Studijní obor:	**3902T004-00 Automatické řízení a inženýrská informatika**
Pracoviště:	**Katedra automatizační techniky a řízení - 352**

Zásady pro zpracování:

1. Proveďte detailní rozbor a charakterizujte podstatu webových služeb a jejich standardů (XML, SOAP, wsdl, disco) v technologii .NET.

2. Analyzujte vlastnosti sériové komunikace nejprve obecně a poté možnosti komunikace v technologii .NET s využitím jazyka C#.

3. Popište I/O Jednotku LabJack USB, charakterizujte možnosti jejího využití a navrhněte a realizujte podporu pro laboratorní úlohu se vstupně výstupní jednotkou.

4. Pro realizovanou úlohu vytvořte webovou službu v technologii .NET. Realizujte klientskou aplikaci s grafickou podporou konzumující webovou službu.

5. Zhodnoťte dosažené výsledky a přínosy práce, nastiňte možnosti využití a dalšího vývoje v této oblasti.

Pokyny pro zpracování:

Rozsah práce: Minimálně 40 stran včetně příloh

Seznam doporučené literatury:

PAYNE, CH. *Naučte se asp.net za 21 dní, Pr*aha : Computer Press, 2002. 746 s. ISBN 80-7226-605-5.

DUTHIE, A. G. *ASP.NET krok za krokem.* Brno: Knihy.iDnes 2002, 511 s, ISBN 80-86593-33-9.

KAČMÁŘ, D. *Programujeme .NET aplikace ve visual studiu .net,* Praha: 2001.335 s. ISBN 80-7226-569-5.

LACKO, L. *Programujeme mobilní aplikace ve Visual Studiu .NET, Pr*aha : Computer Press, 2004. 480 s. ISBN 80-251-0176-2.

ESPOSITO, D.: *Introducing Microsoft ASP.NET 2.0* , Microsoft Press Redmond, Washington 2005, 427 s. ISBN: 0-7356-2024-5.

PROSISE, J.: *Programování v Microsoft .NET.* Computer Press, 2003, 736 s., ISBN 80-7226-879-1.

Vedoucí diplomové práce:	Ing. Marek Babiuch, Ph.D.
Datum zadání diplomové práce:	6. 12. 2008
Termín odevzdání diplomové práce:	23. 5. 2009
Akademický rok:	**2008/2009**

L. S.

.....	.....
prof. Dr. RNDr. Lubomír Smutný	**prof. Ing. Radim farana, CSc.**
vedoucí katedry	děkan Fakulty strojní

V Ostravě dne: 5. 11. 2008

Prohlášení diplomanta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě …….. ……

Martin Hnik

Prohlašuji, že

- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§35 odst. 3).
- souhlasím s tím, že jeden výtisk diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o diplomové práci, obsažené v Záznamu o závěrečné práci, umístěném v příloze mé diplomové práce, budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu §12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě …… ……
Martin Hnik

Martin Hnik
E. Rošického 1074/6
721 00, Ostrava – Svinov

ANOTACE DIPLOMOVÉ PRÁCE

HNIK, M. *Využití .NET webových služeb v prostředí automatizace.* Ostrava: katedra automatizační techniky a řízení, Fakulta strojní VŠB-Technická univerzita Ostrava, 2009, 50 s. Diplomová práce, vedoucí: Babiuch, M.

Tato práce se zabývá technologií pro výměnu dat XML Web Services a jejím uplatnění na konkrétních úlohách. Jedna z vytvořených aplikací umožňuje monitorovat a ovládat reálný tepelný proces pomocí mnoha klientských zařízení nezávislých na použitém operačním systému, druhu nebo např. poloze. Tepelný proces tak může být ovládán např. jiným procesem, webovou stránkou nebo mobilním telefonem. Systém je vytvořen od základu a obsahuje tři hlavní části. Hardwarová část je realizovaná pomocí měřicí karty, akčních členů a teplotních senzorů. Jádrem aplikace je server, na kterém běží XML Web Service, Windows Service a SQL Server. Byl vytvořen také klientský software pro mobilní telefony a webové stránky.

ANNOTATION OF THE THESIS

HNIK, M. *Using .NET web Services at area of Automation.* Department of Control Systems and Instrumentation, Faculty of Mechanical Engineering VŠB-Technical University of Ostrava, 2009, 50 p. Thesis, head: Babiuch, M.

This work addresses to uses of the XML Web Service technology through case studies on the development on particular applications. The goal of one case study is to create an application where users can monitor and control a real process via various devices regardless of how the device systems are implemented or where they reside. The device can be for example another real process, a web page or a cell phone. To create such application the work has to deal with several issues. At the first the work has to deal with hardware equipments and the thermal process itself. The application core is an XML Web Service using a MS SQL database for storing and process data. For reason of continuous data measurement there is a Windows Service. The thermal process can be controlled via various devices. Users can see a graph of actual temperature on a cell phone or they can control the thermal process.

Obsah

Seznam použitého značení

API	**Application Programming Interface**, sada nástrojů programu nebo operačního systému pro podporu programování aplikací.
GPRS	**General Packet Radio Service**, způsob přenosu dat v mobilních sítích.
LINQ to SQL	**Language Integrated Query**, integrovaný jazyk pro dotazování nad databázemi v prostředí Microsoft .NET Framework.
Namespace	Jmenný prostor – abstraktní prostředí, které sdužuje logické skupiny názvů.
SMTP	**Simple Mail Transfer Protocol**, internetový protokol určený pro přenos zpráv elektronické pošty.
SOAP	**Simple Object Access Protocol**, protokol pro výměnu zpráv v síti Internet založených na jazyku XML.
UDDI	**Universal Description, Discovery and Integration**, univerzální adresář obsahující seznam a popis dostupných webových služeb.
URL	**Uniform Resource Locator**, definovaná struktura zápisu umístění zdrojů informací na Internetu.
WDDX	**Web Distributed Data Exchange**, technologie výměny dat založená na formátu XML.
WSDL	**Web Services Description Language**, forma popisu webové služby vycházející z jazyka XML.
XML	**eXtensible Markup Language**, obecný rozšiřitelný značkovací jazyk.

Úvod

Tato práce se zabývá možnostmi využití webových služeb jako nástroj pro ovládání a výměnu dat mezi technologickým procesem a klienty, kteří k němu přistupují. Klientem může být například mobilní aplikace, webová stránka ale i jiný technologický proces. Webová služba je univerzální pro všechny typy koncových zařízení.

První část práce je věnovaná teoretickému popisu technologie XML Web služeb. Webová služba je, jak ji definuje současný správce standardu pro přenos dat konsorcium W3C, řešení, jak spolu mohou komunikovat a vyměňovat si informace aplikace přes Internet. Z této definice lze vyčíst univerzálnost tohoto řešení. Aplikace si mezi sebou vyměňují standardizované XML zprávy. Každá webová služba obsahuje svůj standardizovaný popis funkcí a parametrů. Tím je zajištěna možnost použít je v masivním měřítku.

Webovými službami je možné sdílet funkce, možnosti, data, a dokonce i samotné zařízení mezi společnostmi, aplikacemi na úrovni společnosti a v aplikaci samotné. Poslední dvě kapitoly práce se věnují dvěma praktickým ukázkám využití této technologie v prostředí automatizace.

1 Webové služby

Webová služba je nástroj pro komunikaci a výměnu informací v prostředí Internetu. Tento nástroj je otevřený širokému spektru různých aplikací komunikujících mezi sebou.

Klienti webových služeb nemusí dopředu znát detailní popis služby, ale mohou si jej kdykoliv zjistit. Binární forma předávání dat nemusí být na všech různých operačních systémech identická. Naproti tomu webové služby využívají jazyk XML, umožňující předat data zcela jednotně, nezávisle na klientském operačním systému.(Kačmář, 2001)

Obrázek 1.1 XML webová služba komunikuje s různými druhy klientů.

Pokud chceme komunikovat s nějakou webovou službou, stačí poslat prostřednictvím HTTP protokolu příslušnou zprávu XML. Protože každé zařízení, které je určeno pro Internet podporuje HTTP protokol, a protože velká část programovacích jazyků umožňuje přistupovat k XML parseru, snižují se tak omezení ohledně typů aplikací, které mohou používat webové služby. V podstatě většina programovacích prostředí obsahuje sadu nástrojů vyšší úrovně, díky kterým je komunikace s webovou službou stejně snadná jako volání lokální funkce.(MacDonald, a další, 2006)

Zatímco HTML stránky (nebo HTML výstup generovaný dynamickými webovými stránkami jako např. ASP.NET nebo PHP) by měly být užívány koncovými uživateli v jejich prohlížečích, webové služby jsou používány pro komunikaci mezi aplikacemi.

Pomocí webových služeb můžeme poměrně snadno používat obchodní logiku někoho jiného, aniž byste ji museli vytvářet od úplného začátku. Tato technika je podobná technice, kterou používají programátoři v případě knihoven API, tříd a komponent. Hlavní rozdíl je v tom, že webové služby mohou být umístěny na vzdáleném serveru a zpravovány jinou společností.(MacDonald, a další, 2006)

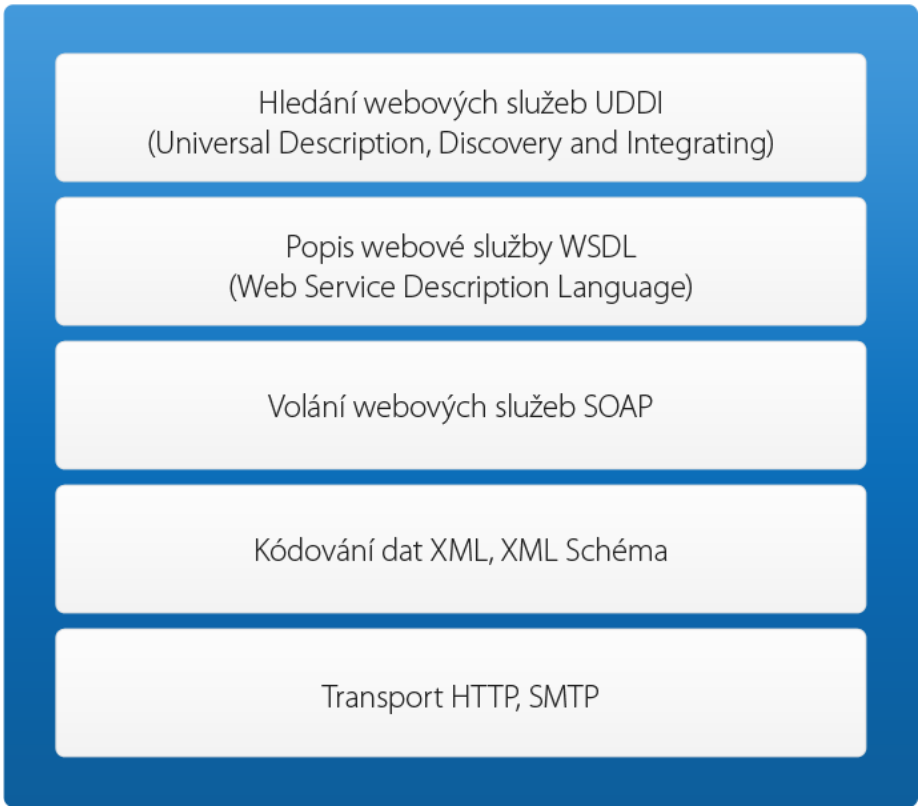
1.1 Historie webových služeb

Dvě z největších změn v oblasti vývoje softwaru, které proběhly během několika posledních dekád, se týkaly vývoje objektově orientovaného programování a technologie založené na komponentách.(MacDonald, a další, 2006)

Objektově orientované programování se připojilo k hlavnímu programovacímu proudu někdy v osmdesátých letech minulého století. Mnozí lidé v objektově orientovaném programování viděli řešení softwarové krize, která vyplývala z narůstající složitosti a velikosti softwarových aplikací. Mnoho projektů se opozdilo a překročilo svůj rozpočet, přičemž výsledný produkt byl často nespolehlivý. Objektově orientovaný kód sliboval začlenění objektů do kódu, díky čemuž mohli vývojáři vytvářet komponenty, které byly opakovaně použitelné, rozšiřitelné a snadno udržovatelné. Zatímco objektově orientovaný jazyk umožňoval vývojářům opakovaně používat objekty ve svých aplikacích, technologie založená na komponentách jim umožňovala snadno sdílet zkompilované objekty mezi aplikacemi. Vznikly dvě dominantní technologie založené na komponentách – ***COM*** (Component Object Model) a ***CORBA*** (Common Object Request Broker Architecture). Od té doby se objevily další technologie založené na komponentách (jako například ***JavaBeans*** a ***.NET***), nicméně jsou navrženy jako proprietární řešení pro konkrétní programovací prostředí.(MacDonald, a další, 2006)

1.2 Standardy používané webovými službami

Klíčem k úspěchu webových služeb je to, že jsou založeny na otevřených standardech, a že za těmito standardy stojí velké společnosti, jako je Microsoft, IBM, Sun a další. Při budování webových služeb se používá několik specifikací:



Obrázek 1.2 Standardy, které dnes používají webové služby.

WSDL	Používá se na vytvoření definice rozhraní webové služby. WSDL dokument oznámí klientovi, které metody jsou ve webové službě přítomny, které parametry a návratové hodnoty jednotlivé metody používají, a jakým způsobem s nimi má komunikovat.
SOAP	Formát zprávy, který je použit pro kódování informací (jako jsou například hodnoty dat) před jejich zasláním webové službě.
HTTP	Prostřednictvím tohoto protokolu probíhá veškerá komunikace webových služeb. SOAP zprávy jsou například zasílány přes HTTP kanály.
DISCO	Používá se k vytvoření tzv. *discovery dokumentů*, které poskytují odkazy na více koncových bodů webové služby. Tento standard je specifický pro Microsoft a posléze bude nahrazen podobným standardem pojmenovaným jako WS-Inspection.
UDDI	Standard pro vytváření obchodních rejstříků, které registrují společnosti a webové služby, které nabízejí, a také odpovídající URL adresy jejich WSDL kontaktů.

Tabulka 1 Standardy webových služeb.

1.3 SOAP

SOAP (celým názvem Simple Object Access Protocol) je protokolem pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP. Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace. SOAP dokument je jednoduchý XML protokol, používající speciální slovník, který popisuje volání funkcí a předávané parametry. SOAP dokumenty pro formátování dotazů a odpovědí lze vytvářet programově a tím se celá práce s webovými službami výrazně zjednodušuje. (Kačmář, 2001)

Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Nejznámější z nich je ***RPC šablona***, kde jeden z účastníků komunikace je klient a na druhé straně je server. Server ihned odpovídá na požadavky klienta. SOAP je nástupce za ***XML-RPC***, ačkoli si zapůjčuje způsob přenosu dat a další vlastnosti. Obálka, hlavička a tělo komunikace je ale pravděpodobně z WDDX.(Wikipedia, 2007)

Původně ho navrhl Dave Winer, Don Box, Bob Atkinson a Mohsen Al-Ghosein v roce 1998 za podpory firmy Microsoft, kde tou dobou Atkinson a Al-Ghosein pracovali. Dnes je SOAP specifikace držena XML skupinou tvořící internetové protokoly z W3C konsorcia. (Wikipedia, 2007)

1.4 WSDL

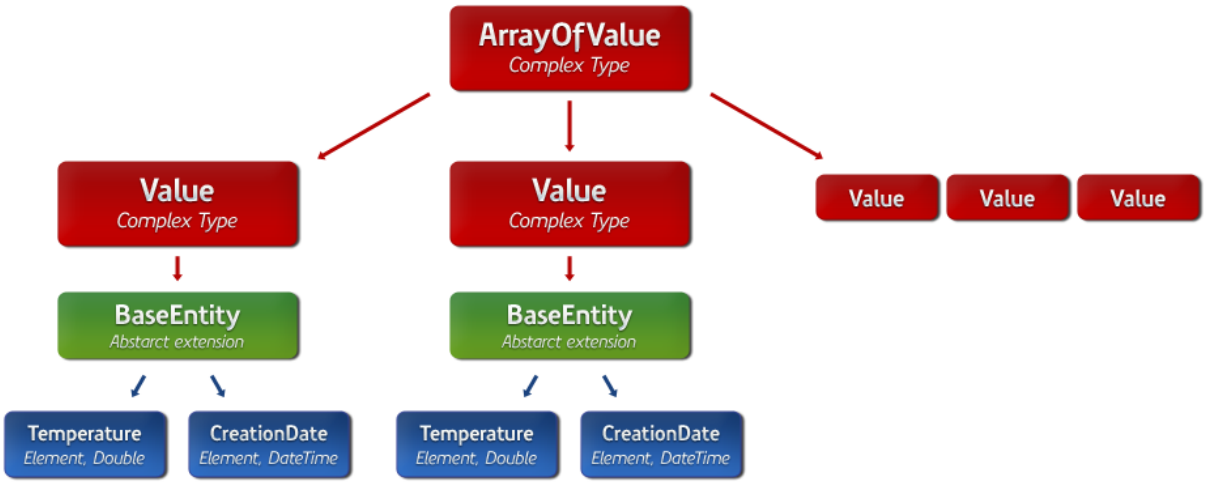
Jak už je napsáno výše, WSDL dokument je forma popisu webové služby vycházející z jazyka XML. WSDL dokument popisuje samotnou webovou službu, její funkce, návratové hodnoty a způsob jakým můžeme s webovou službou komunikovat. Protože jsou webové služby určeny primárně ke sdílení obchodní logiky mezi různými k sobě nijak nesouvisejícími společnostmi resp. aplikacemi, WSDL dokument je často jediným zdrojem informací.

1.5 Ukázka WSDL dokumentu

Tento WSDL dokument popisuje mnou vytvořenou webovou službu k ovládání zařízení LabJack UE9. Služba je popsána v kapitole 5 Systém sledování teploty. Následující dokument WSDL je značně zkrácený.

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
    <wsdl:types>
        <s:schema>
            <s:element name="GetTemperaturesResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="GetTemperaturesResult" type="tns:ArrayOfValue" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ArrayOfValue">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded" name="Value" nillable="true" type="tns:Value" />
                </s:sequence>
            </s:complexType>
            <s:complexType name="Value">
                <s:complexContent mixed="falvse">
                    <s:extension base="tns:BaseEntity">
                        <s:sequence>
                          <s:element minOccurs="1" maxOccurs="1" name="Temperature" type="s:t" />
                          <s:element minOccurs="1" maxOccurs="1" name="CreationDate" type="s:dateTime" />
                        </s:sequence>
                    </s:extension>
                </s:complexContent>
            </s:complexType>
            <s:complexType name="BaseEntity" abstract="true" />
            <s:element name="ArrayOfValue" nillable="true" type="tns:ArrayOfValue" />
        </s:schema>
    </wsdl:types>
    <wsdl:message name="GetTemperaturesSoapOut">
        <wsdl:part name="parameters" element="tns:GetTemperaturesResponse" />
    </wsdl:message>
    <wsdl:portType name="WebMSoap">
        <wsdl:operation name="GetTemperatures">
            <wsdl:input message="tns:GetTemperaturesSoapIn" />
            <wsdl:output message="tns:GetTemperaturesSoapOut" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:service name="WebM">
        <wsdl:port name="WebMSoap" binding="tns:WebMSoap">
            <soap:address location="http://78.108.144.246:4008/WebService.asmx" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

V první části zkráceného dokumentu WSDL jsou definovány elementy. Pro ukázku jsem zvolil jen návratový element GetTemperaturesResult metody GetTemperatures, která vrací n posledních hodnot aktuální teploty. Na následujícím obrázku je popsáno schéma návratového XML dokumentu resp. objektu GetTemperaturesResult.



Obrázek 1.3 Schéma XML elementů metody GetTemperaturesResult

Z dokumentu WSDL lze vyčíst mnoho informací. Metoda *GetTemperatures* vrací kolekci objektů *Value*. Tato kolekce je nazvána ArrayOfValue. Třída Value je potomek abstraktní třídy *BaseEntity*. Třída Value obsahuje mimo jiné dvě vlastnosti Temperature a CreationDate.

1.6 Formy přenosu

Webové služby komunikují pomocí standardních protokolů HTTP nebo SMTP. Jinými slovy HTTP nebo SMTP tvoří aplikační vrstvu komunikace. Mezi rozšířenější patří protokol HTTP. 0ba tyto protokoly jsou navrženy pro práci v prostředí Internetu a proto jejich porty jsou otevřeny na většině standardních firewallů a díky tomu může *SOAP jednoduše procházet přes firewall*. Toto je velká výhoda webových služeb.(Manes, 2003)

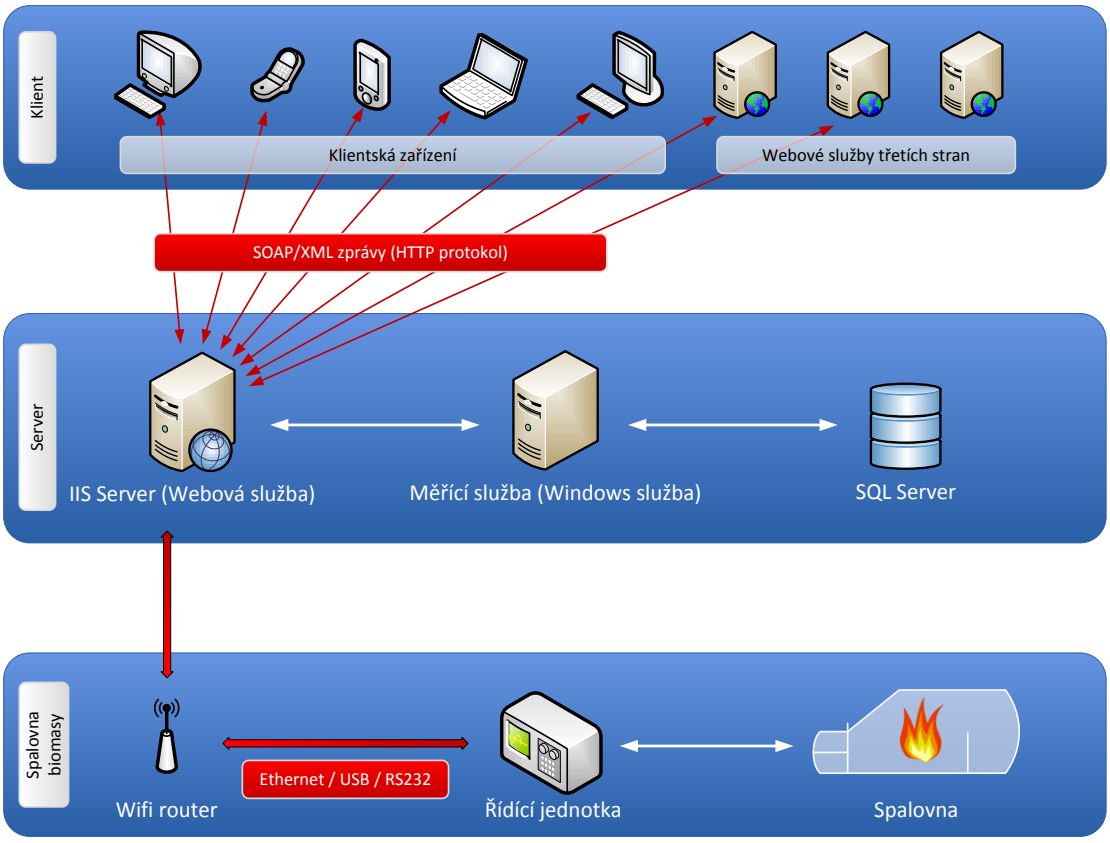
Zdlouhavá syntaxe XML má své výhody i nevýhody. Je jednoduše čitelná pro člověka, ale počítač ji musí složitě parsovat a stojí to hodně procesorového času a operační paměti. Binární způsob komunikace má zápis zpráv daleko kratší, ale je pro člověka nečitelný. Na druhou stranu vývoj počítačů jde rychle dopředu a přestává to být na obtíž. Byla již vytvořena i binární forma XML.(Wikipedia, 2007)

1.7 Použití webových služeb v prostředí automatizace

Síla webových služeb je především v jejich univerzálnosti a nezávislosti na koncovém zařízení. Rychlost odezvy webové služby je přímo úměrná rychlosti

odezvy mezi koncovým klientem a serverem a rychlostí přenosu XML dokumentu a tedy objemem přenášených dat. Díky tomu, že formát XML je vytvořen podobně jako textový formát, přenáší se větší objem dat než u přenosu souboru v binárním formátu. Nevýhodou může tedy oproti jiným řešením být přenos většího objemu dat. Díky pokroku v rychlosti přenosu dat se tato nevýhoda stává méně podstatnou.

Tato technologie najde uplatnění v technologických procesech a měřeních, kde nevyžadujeme od systému odezvu v řádu milisekund, ale dostatečné jsou odezvy v řádu sekund nebo i desítek sekund. Formát XML umožňuje také uchovávat binární soubory a tedy i obraz. Typickým příkladem použití pak může být například systém pro sběr dat z výrobní linky, sledování linky pomocí webové kamery, měření teploty v provozu, ovládání spalovny biomasy apod. Příklad schéma jednoduché aplikace webové služby v provozu může být následující:



Obrázek 1.4 Příklad zjednodušeného schéma aplikace webové služby pro spalovnu biomasy

2 Analýza komunikace s využitím jazyka C#.

Tato kapitola pojednává o způsobech komunikace s využitím jazyka C#. Úvodní část je věnována popisu standardů komunikace a v dalších bodech jsou některé standardy aplikovány na možnosti a omezení jazyku C# resp. .Net Frameworku.

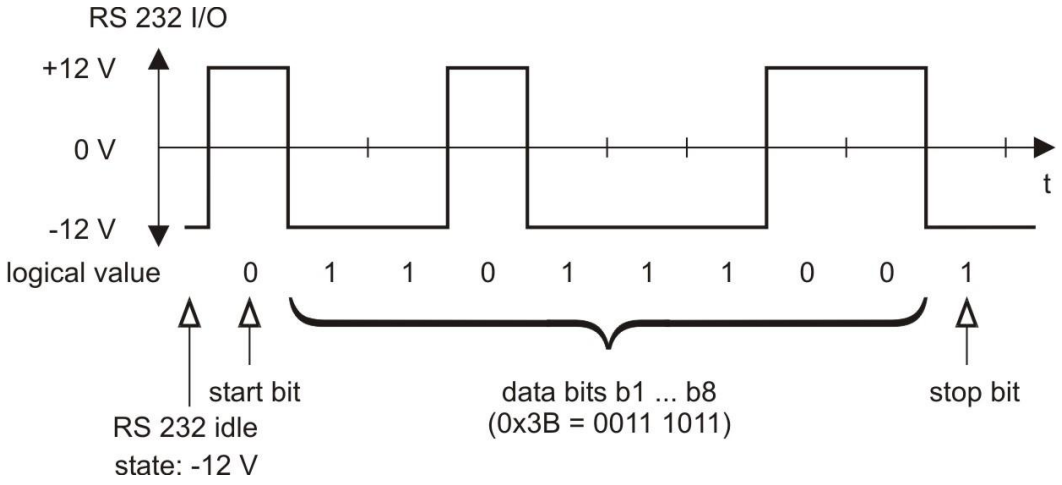
2.1 RS 232

RS-232 je standardní rozhranní pro sériovou komunikaci. Tento standart byl definován dle Electronic Industries Association (EIA). RS-232 existuje ve třech různých verzích A, B a C. Každá z těchto verzí má jiné rozmezí napětí pro *On* a *Off* stavy. Nejvíce využívaný je standart RS-232C, který definuje stav *On* jako napětí mezi -3V a -12V a stav *Off* jako napětí v rozsahu +3V a +12V. Specifikace RS-232C říká, že signál je stabilní, pokud je vysílán po kabelu s maximální délkou 8m. Při komunikaci jsou jednotlivé bity přenášeny postupně za sebou (v sérii) po jediném vodiči podobně jako u síťové technologie Ethernet nebo rozhranní USB.(Cohen, 2005)

2.2 Technický popis RS 232

Sériový port používá asynchronní přenos dat. Při asynchronním přenosu obě zařízení, tedy jak přijímací, tak i vysílací strana, se nejprve musí vhodným způsobem nakonfigurovat, aby přijímač věděl, v jakém formátu má data očekávat a jak rychle je nutné vzorkovat datovou linku, neboli jaká je přenosová rychlost. Na obou zařízeních se musí nastavit počet přenášených bitů v jednom celku, přenosová rychlost uváděná v bitech za sekundu (bitrate) a délka stopbitu. Toto nastavení musí být na přijímači i vysílači shodné. Datový vodič se před vlastním přenosem nachází v klidovém stavu, což znamená vysokou úroveň napětí. Před začátkem odeslání bitů (podle konfigurace) je nejprve poslán takzvaný start bit, který má vždy nulovou hodnotu, tj. je zaručeno, že se klidový stav linky (vysoká úroveň) vždy změní. Na straně přijímacího zařízení se musí co nejpřesněji rozeznat změna stavu linky – sestupná hrana, která reprezentuje začátek start bitu. Od této chvíle přijímací strana ví, že se přijme předem daný počet bitů s předem nastavenou bitovou rychlostí. Přijímací strana tedy vzorkuje stav linky a podle svých vlastních hodin rozeznává hodnoty jednotlivých bitů. Na konci přenášené sekvence může být (v závislosti na nastavené

konfiguraci) přenesen i paritní bit, za nímž ihned následuje takzvaný stop bit. Ten má vždy hodnotu logické jedničky, což znamená vysokou úroveň napětí. Délka stop bitu může být delší než délka ostatních bitů (například lze použít délku rovnou 1,5 násobku či dvojnásobku délky běžného bitu), aby měl přijímač dostatek času na zpracování došlých dat.(Tišnovský, 2008)



Obrázek 2.1 Sériové spojení RS 232(Rankl, 2004)

2.3 Komunikace po RS 232 v .Net Frameworku

Pro komunikaci standardem RS 232 existuje v .Net Frameworku namespace System.IO.Ports. Tento jmenný prostor obsahuje třídy pro práci se sériovým portem. Nejdůležitější třída, SerialPort, umožňuje synchronní komunikaci, přístup k jednotlivým pinům a má přístup k vlastnostem ovladače. (Microsoft Corporation)

Kompletní popis tohoto jmenného prostoru pro komunikaci po standardu RS 232 je k nalezení na http://msdn2.microsoft.com/en-us/library/system.io.ports.aspx. Ukázková aplikace je k nalezení na http://www.codeproject.com/csharp/SerialCommunication.asp.

2.4 Universal Serial Bus

USB (Universal Serial Bus) je univerzální sériová sběrnice. USB byl vytvořen jako průmyslový standard pro architekturu PC se zaměřením na periferie. Následující kritéria byla aplikována v definici pro USB:

- jednoduchá rozšiřitelnost PC
- plná podpora pro *full-time* přenos dat (zvuk, video apod.)
- flexibilita mixování izochronního přenosu dat a asynchronního přenosu zpráv

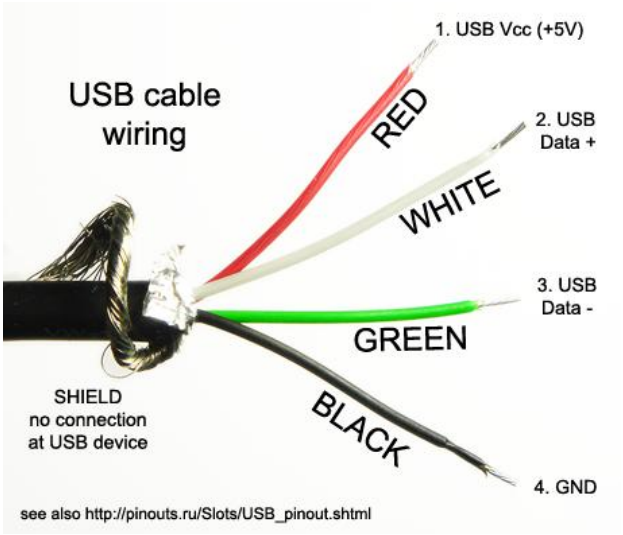
Výhodou USB je možnost připojování periférií bez nutnosti restartování počítače nebo instalování ovladačů. Maximální délka kabelu mezi dvěma zařízeními je 5 metrů. USB dovoluje připojit až 127 zařízení pomocí jednoho typu konektoru. Připojeným zařízení USB zároveň poskytuje i napájecí napětí.(USB Implementers Forum, Inc., 2000)

2.5 Technický popis USB

USB rozhraní používá dva typy konektorů. Plochý konektor „typ A“ je dnes obsažen na prakticky každém novém PC v minimálně 2 konektorech. Druhý konektor „typ B“ je určen pro periferní zařízení, čímž je zároveň definován standard propojovacího kabelu. (Řehák, 2002)

Obrázek 2.2 Základní konektory rozhranní USB, typ A a typ B (KRCS)

Pro komunikaci může být kabel jak stíněný, tak i nestíněný. V kabelu jsou čtyři následující vodiče:



Obrázek 2.3 Kabel pro komunikaci rozhranním USB (Beale, 2007)

Popis jednotlivých pinů je následovný:

Pin	*Jméno*	*Barva*	*Popis*
1	VBus	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	GND	Black	Ground

Existuje několik módů rychlosti:

- High Speed - 480Mbits/s
- Full Speed - 12Mbits/s
- Low Speed - 1.5Mbits/s (Peacock, 2002)

2.6 Komunikace standardem USB v .Net Frameworku

Microsoft. Net Framework nenabízí žádné univerzální rozhranní pro komunikaci standardem USB a proto je nutné použít ovladač k danému zařízení a přistupovat k ovladači nebo přistupovat k Win32 API.

Pro seznámení se s komunikací po USB jsem vytvořil jednoduchou webovou službu, která komunikuje s webovou kamerou. Pro přístup k webové kameře je použit standardní Windows knihovna Windows Image Acquisition DLL, známější jako WIA.

Windows Image Acquisition (WIA)

WIA je standardizované Win32 API pro pořizování snímků ze zařízení jako jsou web kamery, scannery apod. Toto API je dostupné v operačním systému Windows od verze Windows Millennium a je dostupné i v novějších verzích operačního systému Windows v aktualizovaných verzích.(Scheidegge)

Více informací o tomto API je k nalezení na http://msdn2.microsoft.com/en-us/library/ms630368.aspx.

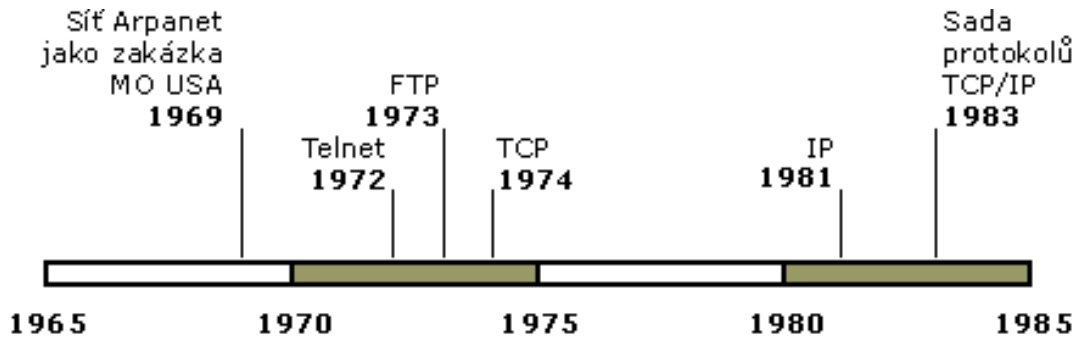
Webová služba přistupující k webkameře

Webová kamera nesnímá video, jak by se mohlo na první pohled zdát, ale snímá jednotlivé snímky v sekvenci za sebou. Principu snímání jednotlivých snímků využívá i má jednoduchá webová služba.

Webová služba přistupuje k Win23 API WIA a používá její metodu *GrabFrame*, která z nakonfigurovaného zařízení sejme snímek. Tento snímek představuje datový proud (stream), který dále poskytuje pomocí protokolu SOAP.

2.7 Síťová komunikace a protokol TCP/IP

Technologie TCP/IP (Transmission Control Protocol/Internet Protocol) je standardní sada protokolů navržená pro použití v rozsáhlých strukturách propojených sítí, které spojují různá prostředí LAN a WAN. Jak ukazuje následující časová osa, počátky protokolu TCP/IP spadají do roku 1969, kdy vznikla zakázka ministerstva obrany USA na vytvoření sítě ARPANET (Advanced Research Projects Agency Network).(Microsoft Corporation, 2005)



Obrázek 2.4 Počátky protokolu TCP/IP (Microsoft Corporation, 2005)

2.8 Technický přehled protokolu TCP

Protokol TCP (Transmission Control Protocol) je standard sady protokolů TCP/IP definovaný ve specifikaci RFC 793, Transmission Control Protocol (TCP), který poskytuje spolehlivé služby doručování paketů orientované na připojení. Protokol TCP má následující možnosti:

- Zaručuje dodání datagramů IP na místo určení.
- Provádí segmentaci a zpětné sestavování velkých bloků dat odeslaných programy.
- Zaručuje doručení segmentovaných dat ve správném pořadí.
- Provádí kontrolu integrity přenášených dat pomocí kontrolního součtu.
- Odesílá zprávy informující o úspěšném přijetí dat. Při použití výběrového potvrzování jsou odesílány také zprávy o datech, která nebyla přijata.
- Nabízí preferované metody přenosu dat programům, které musí využívat spolehlivého přenosu dat na bázi relací, jako jsou například databáze se strukturou klient/server a e-mailové programy. (Microsoft Corporation, 2005)

2.9 Síťová komunikace v .Net Frameworku

Pro síťovou komunikaci disponuje Microsoft .Net Framework jmenným prostorem **System.Net.Sockets**. Tento jmenný prostor umožňuje a řídí práci se sockety pro síťovou komunikaci. (Microsoft Corporation)

Více informací o tomto jmenném prostoru je na http://msdn2.microsoft.com/en-us/library/system.net.sockets.aspx.

Při síťové komunikaci jsou nejdůležitější dvě třídy: **TcpListener** a **TcpClient**. TcpListener slouží k naslouchání na příchozí připojení klientů a TcpClient, která slouží k vytvoření klienta, který se připojuje k posluchači. Klient i posluchač musí komunikovat na stejném portu.

3 Měřicí karta LabJack UE9

http://obchod.hw.cz/DetailPage.asp?CD=900742LabJack UE9 je měřicí karta o rozměrech 188 x 75mm a výšce 31mm. Pro komunikaci s nadřízeným PC obsahuje konektor USB typu B a 10-T Ethernet konektor. LabJack UE9 má možnost napájení přes USB, z externího zdroje přes 2,1 POWER JACK anebo přes 5mm šroubovací svorkovnici. (HW.cz, 2005)

3.1 Popis konektorů

Šroubovací konektory jsou rozděleny do šesti segmentů.

Číslo	Název	Popis	Číslo	Název	Popis
1. Segment			4. Segment		
1	AIN0	Analogový vstup	13	FIO0	Časovač, čítač
2	AIN1	Analogový vstup	14	FIO1	Časovač, čítač
3	GND	Zem	15	GND	Zem
4	VSS	Napájecí napětí 5V	16	VSS	Napájecí napětí 5V
2. Segment			5. Segment		
5	AIN2	Analogový vstup	17	FIO0	Časovač, čítač
6	AIN3	Analogový vstup	18	FIO1	Časovač, čítač
7	GND	Zem	19	GND	Zem
8	VSS	Napájecí napětí 5V	20	VSS	Napájecí napětí 5V
3. Segment			6. Segment		
9	DAC0	Analogový výstup	21	SCA	Pro budoucí použití
10	DAC1	Analogový výstup	22	SCL	Pro budoucí použití
11	GND	Zem	23	GND	Zem
12	VSS	Napájecí napětí 5V	24	VSS	Napájecí napětí 5V

Tabulka 2 Šroubovací konektory zařízení LabJack UE9

Mezi hlavní vlastnosti měřicí karty patří:

- 14 analogových vstupů (12–16Bit),
- ±5 nebo 0–5 V maximální vstupní napětí,
- 2 analogové výstupy (12Bit, 0–5V),

- 23 digitálních I/O,
- nejnižší odezva 1.2ms,
- podpora USB 2.0/1.1,
- ethernet 10Base-T rozhranní,
- rozměry 75mm x 185mm x 30mm,
- provozní teploty od -40 do +85 °C.(LabJack Corporation)

Obrázek 3.1 Měřicí ústředna LabJack UE9

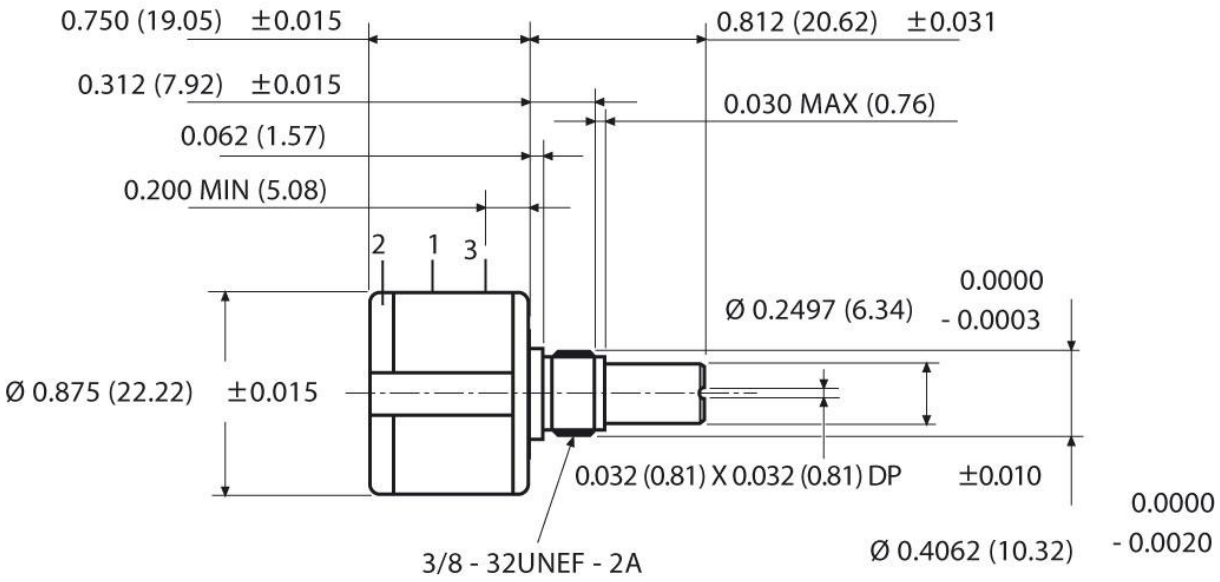
4 Měření úhlu natočení pomocí zařízení LabJack UE9

Úloha slouží k zjištění úhlu natočení. Úhel natočení je snímán odporovým snímačem v potenciometrickém (s paralelním voltmetrem) zapojení.

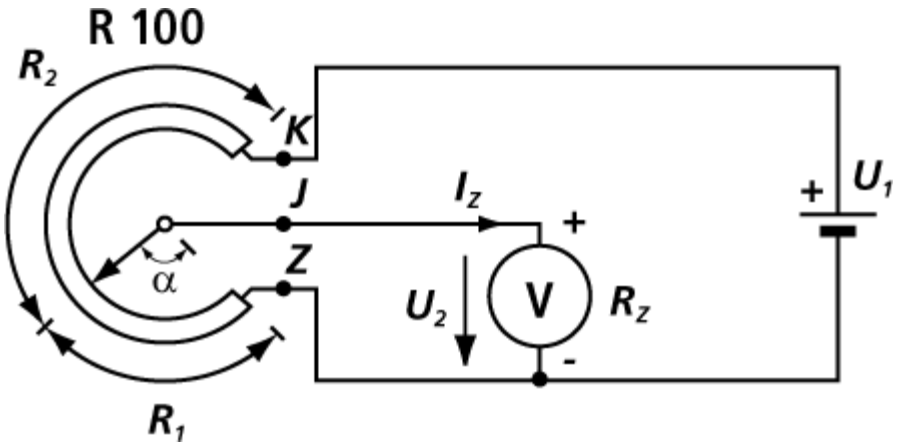
4.1 Použitý snímač natočení

Jako snímač úhlu natočení jsem vybral a zakoupil víceotáčkový potenciometr PM-534 100R s těmito vlastnostmi:

- R = 0 – 100 Ω
- Tolerance na odpor 1K ± 5%
- Tolerance na linearitu ± 0,25%
- Úhel natočení 3600° + 10%



Obrázek 4.1 Víceotáčkový potenciometr PM-534 100R (Vishay Spectrol, 2004)



Obrázek 4.2 Schéma zapojení odporového snímače natočení

V tomto zapojení je napětí lineární funkcí změny odporu. Technicky měříme výstupní napětí a přepočtem podle vztahu [1] získáme úhel natočení.

$$U_2 = U_1 \frac{R_1}{R_1+R_2} = U_1\alpha \qquad [1]$$

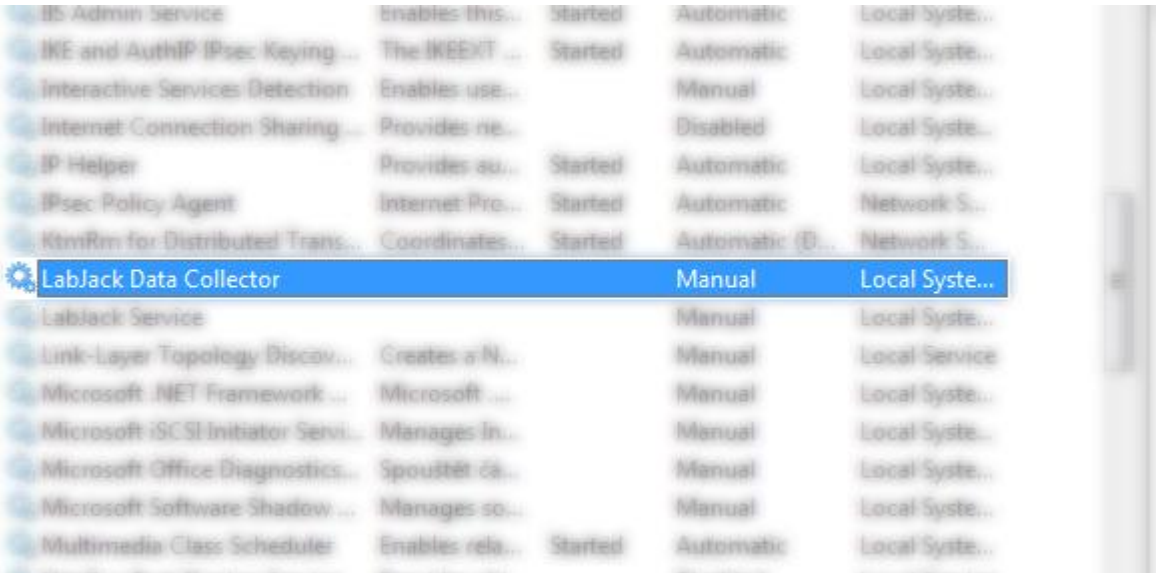
4.2 Technické zpracování

V této kapitole se budu zabývat samotným technickým zpracováním, tvorbou služby Windows, XML webové služby, klienta pro mobilní zařízení a klienta pro webový prohlížeč.

Myšlenka měření je taková, že se budou naměřená data ukládat v pravidelném intervalu do relační databáze. Pomocí webové služby se pak tato data zprostředkují klientům v různých formách jako např. naměřená hodnota napětí, úhel natočení, úhel natočení v minutách a vteřinách apod. Klienti přistupující k této webové službě obdrží množství dat, o které si zažádají. Např. mobilní klient nebude potřebovat stovky posledních uložených hodnot, ale požádá si jen o posledních 10. Naopak webový klient nemá tak velká omezení datového přenosu jako mobilní a může si zažádat o více dat.

Windows service

Vytvořil jsem Windows service pojmenovaný jako LabJack Data Collector, který komunikuje se zařízením v určitém intervalu (3s), sbírá ze zařízení aktuální hodnotu měřeného napětí U_2 a ukládá ji do MS SQL databáze pro pozdější zpracování. Při vytváření Windows service byl vytvořen i jeho instalační balíček.



Obrázek 4.3 Nainstalovaná služba LabJack Data Collector

Webová služba

Vytvořil jsem webovou službu pojmenovanou LabJackWebService, která poskytuje jak přístup do databáze naměřených hodnot, tak i dokáže komunikovat přímo se zařízením.

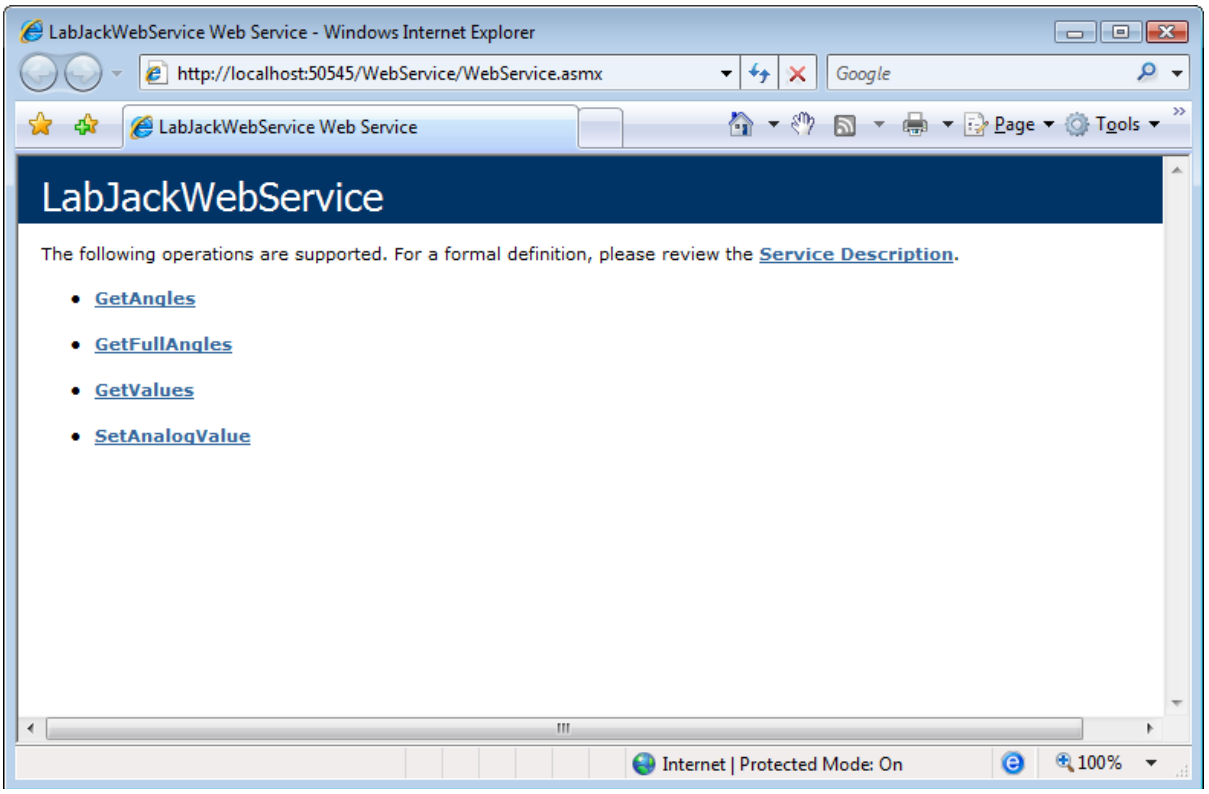
Webová služba obsahuje čtyři metody:

GetAngles : získá z databáze posledních N naměřených hodnot úhlu. Úhly jsou od sebe odděleny středníkem.

GetFullAngles: získá z databáze posledních N přesnějších naměřených hodnot úhlu. Úhly jsou včetně minut a vteřin. Úhly jsou od sebe odděleny středníkem.

GetValues: získá z databáze posledních N naměřených hodnot napětí. Hodnoty jsou od sebe odděleny středníkem.

SetAnalogValue: nastaví na požadovaném analogovém výstupu požadovanou hodnotu napětí v rozmezí 0–5V. Tato metoda slouží například pro zapnutí světla.



Obrázek 4.4 Výchozí stránka webové služby s odkazem na WSDL dokument

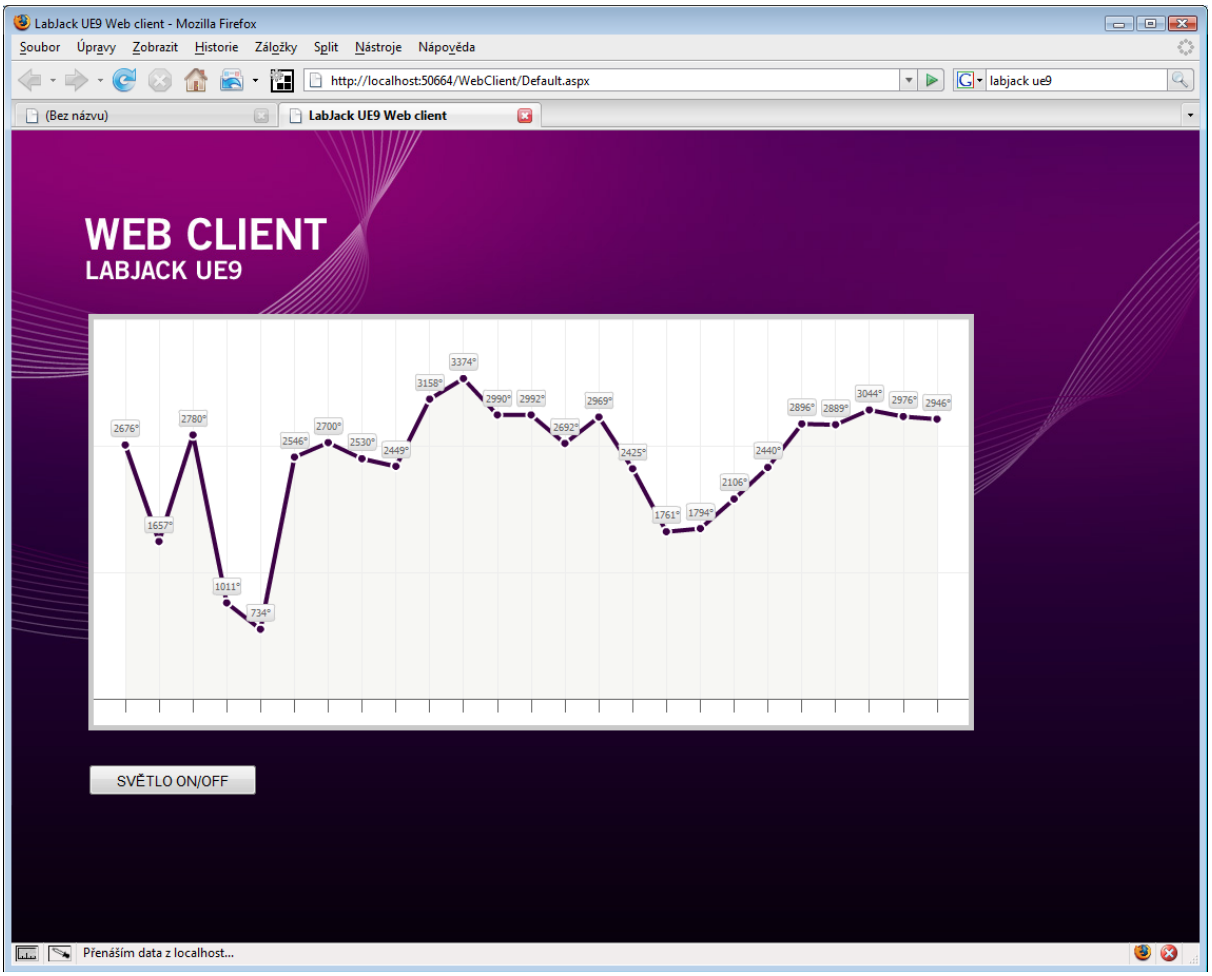
4.3 Webová stránka jako klient

Největší část práce jsem věnoval samotnému zobrazování dat. K tomuto účelu jsem vytvořil svou vlastní komponentu pro vykreslování dynamických dat.

Komponenta je napsána programovacím jazykem ActionScript v prostředí Adobe Flash a zpracovává vstupní údaje z jiné stránky nebo i ze své vlastní „hostitelské“. Komponenta je schopna konzumovat i samotné webové služby, ale tento způsob byl velmi pomalý a zpracování trvalo i několik sekund, což bylo neúnosné pro zobrazování dynamických dat. Praktické měření ukázalo odezvu až 10s.

Zpracování dat probíhá tedy tak, že konzumentem webové služby je ASP.NET webová stránka resp. aspx soubor, který zpracovaná webovou službu a posílá data komponentě v nesrovnatelně kratším čase, než to dokáže Flash.

Webová stránka komunikuje také prostřednictvím webové služby se samotným zařízením a dokáže zapnout nebo vypnout světlo připojené k zařízení LabJack UE9.

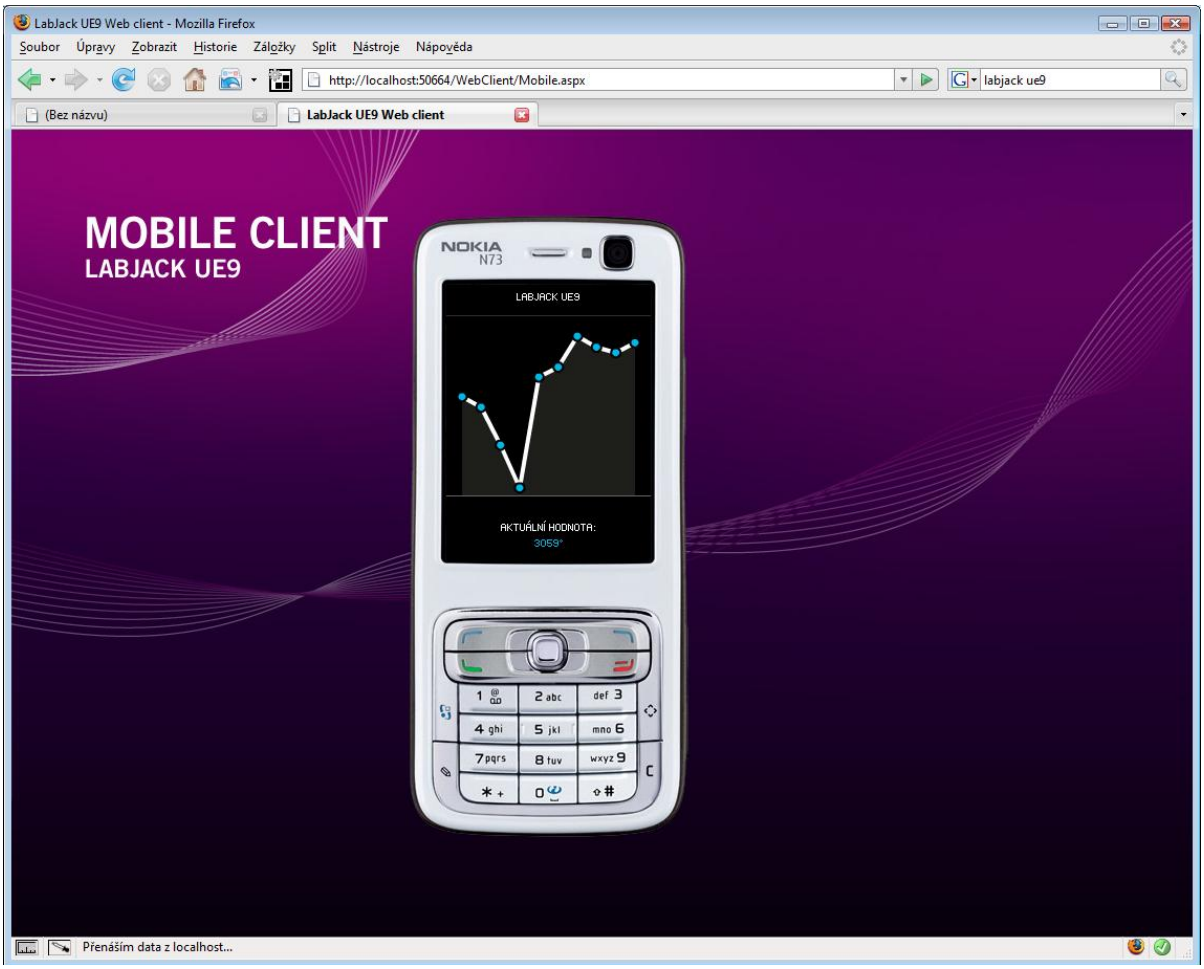


Obrázek 4.5 Klient jako webová stránka

4.4 Klient pro mobilní zařízení

Protože podstata webových služeb je právě v jejich univerzálnosti nasazení, rozhodl jsem se vytvořit i klienta pro mobilní zařízení.

Klient je založen na Adobe Flash komponentě, která je použita výše. Je však značně upravena a podřízena omezením při použití v mobilních zařízeních. Omezeními jsou myšleny horší zobrazovací schopnosti, omezený výpočetní výkon a hlavně omezená rychlost přenosu dat. Aplikace je přizpůsobena pro mobilní zařízení s připojením přes GPRS a vyžaduje technologii Adobe Flash Lite 2.0, která je dostupná ve všech novějších telefonech střední a vyšší třídy.



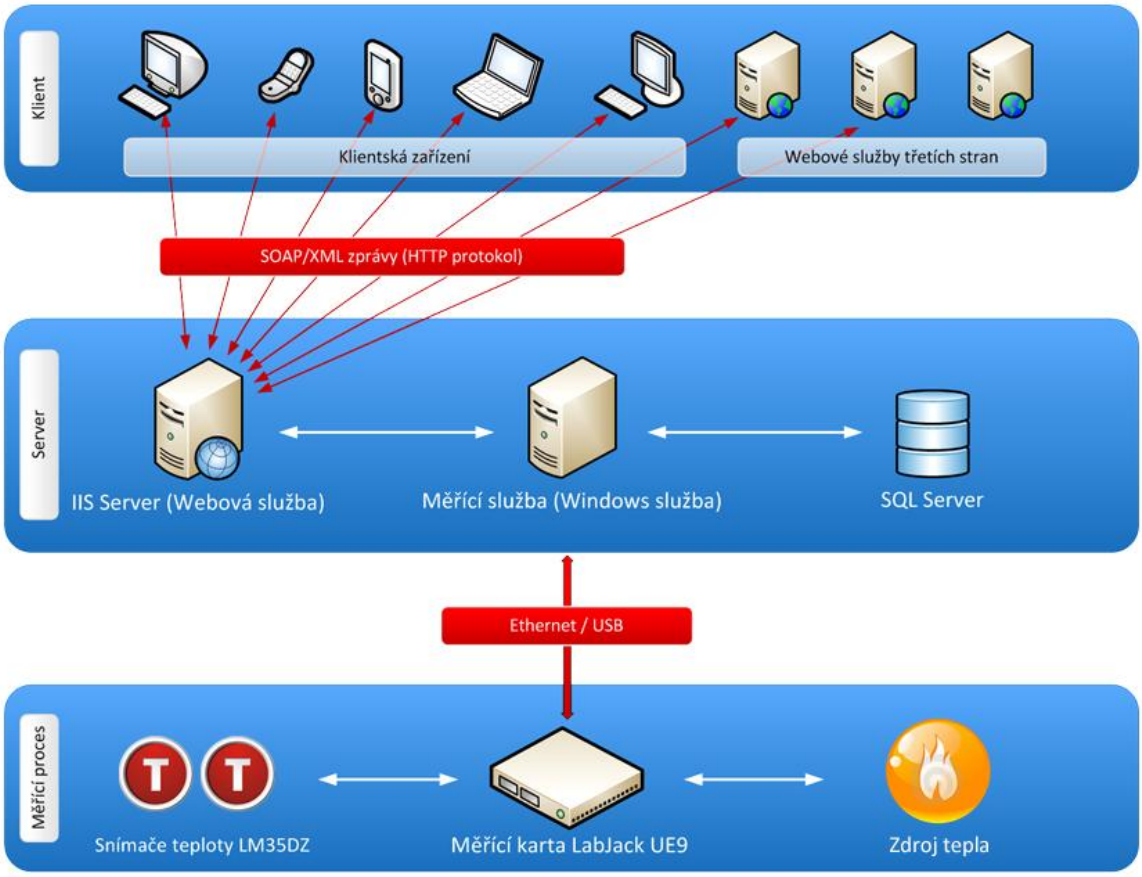
Obrázek 4.6 Ukázka mobilního klienta

5 Systém sledování teploty

Výsledkem úlohy systému sledování teploty je aplikace, která data změří, vyhodnotí, uloží do databáze a pomocí webové služby poskytuje tato data dále klientským aplikacím. Použitím webové služby aplikace zajistí bezproblémovou komunikaci mezi všemi možnými druhy klientů. Součástí úlohy je také návrh samotných teploměrů pro získávání měřených hodnot a klient pro mobilní zařízení.

5.1 Popis systému

Navržený systém se dá rozdělit do několika dílčích úloh, které v celku přináší ucelené řešení. Následující schéma zobrazuje strukturu navržené systému.



Obrázek 5.1 Schéma měřicího systému

Měřicí proces

Úkolem této vrstvy je snímání analogových veličin. V tomto případě se jedná a o snímání teploty pomocí čtyř analogových teploměrů LM35DZ (viz. 5.2 Technické zpracování měření). Pomocí zapojení teploměru (viz. Obrázek 5.2 Schéma zapojení teploměru) získáme lineární závislost měřené teploty na výstupním napětí. Zapojení je napájeno měřicí kartou. Výstupní napětí je dále snímáno měřicí kartou LabJack UE9. Karta může komunikovat se serverem hned několika možnostmi a to pomocí USB rozhraní nebo využitím ethernetu. Výhodnější variantou pro tento systém je použití ethernetu a to hlavně díky možnosti výrazně větší vzdálenosti mezi měřicí kartou a serverem.

Server

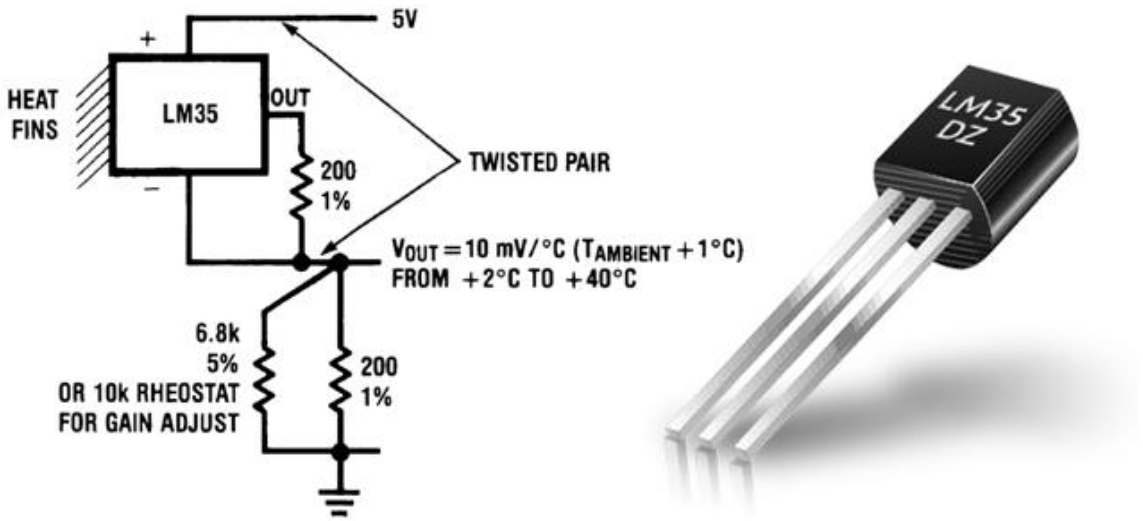
Na serveru probíhá zpracování, ukládání a poskytování dat klientům pomocí webové služby. Pro tyto účely byla vytvořena solution s názvem WebM (viz. 5.3 Řešení na straně serveru) zahrnující aplikace pro komunikaci s SQL Serverem, službu systému Windows, aplikaci pro jednoduchou správu Windows služby, instalační balíček a webovou službu. Výhoda pouze jedné solution je ve snadné rozšiřitelnosti celého systému. To je díky tomu, že jednotlivé aplikace sdílejí mezi sebou společné části a každá tato část existuje v solution pouze jednou. Pokud by bylo třeba systém změnit např. pro jinou měřicí kartu, stačí změnit pouze jednu třídu v celé solution a ostatní části se změna nijak nedotkne. Aplikace na straně serveru (WebM) je vytvořena pomocí Microsoft .Net Framework v jazyce C#.

Klientská část

V této vrstvě systému probíhá již jen zobrazování naměřených dat, popřípadě jejich další zpracování jinými webovými službami. Klienti mohou komunikovat se serverem pomocí webové služby použitím HTTP nebo SMTP protokolu. Součástí systému je i ukázkový klient pro mobilní a přenosná zařízení (viz. 5.4 Klientská aplikace pro mobilní zařízení).

5.2 Technické zpracování měření teploty

Měření je zrealizováno pomocí čtyř teploměrů LM35DZ zapojených pomocí následujícího schéma.



Obrázek 5.2 Schéma zapojení teploměru LM35DZ.(National Semiconductor, 2000)

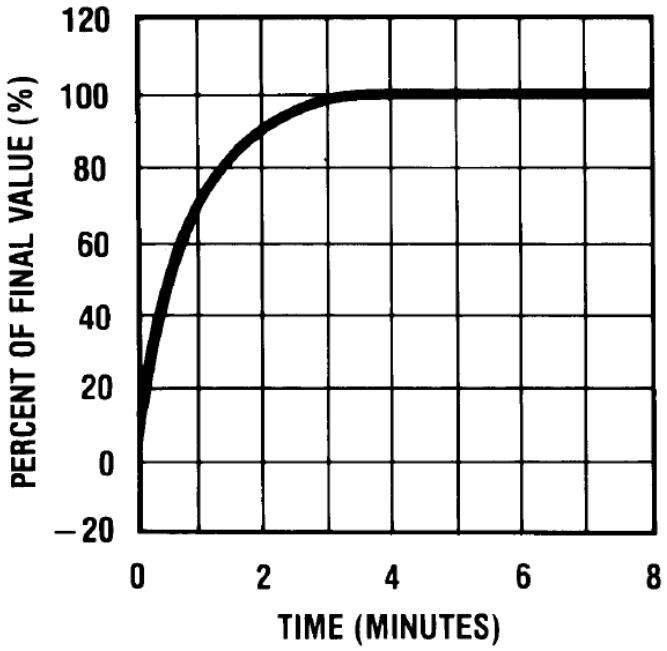
Zapojení je vhodné pro přenos na větší vzdálenosti a relativně imunní proti zkreslení hodnot způsobené nepříznivým vlastnostem přívodního vodiče a okolního rušení.(National Semiconductor, 2000).

Výstupní napětí je lineární závislostí na teplotě s přesností ±0.25°C na linearitu. Teploměr je kalibrován pro výstup ve stupních Celsia. Mezi další důležité vlastnosti patří:

- lineární měřítko je + 10.0 mV/°C,
- přesnost měření při +25°C je 0.5°C,
- typická nelinearita je ±0.25°C.

Parameter	Conditions	LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C	T_A=+25˚C	±0.4	±1.0		˚C
	T_A=−10˚C	±0.5		±1.5	˚C
	$_A=T_{MAX}$	±0.8		±1.5	˚C
	$T_A=T_{MIN}$	±0.8		±2.0	˚C
Accuracy, LM35D	T_A=+25˚C	±0.6	±1.5		˚C
	$T_A=T_{MAX}$	±0.9		±2.0	˚C
	$T_A=T_{MIN}$	±0.9		±2.0	˚C
Nonlinearity	$T_{MIN}\leq T_A\leq T_{MAX}$	**±0.2**		**±0.5**	˚C
Sensor Gain (Average Slope)	$T_{MIN}\leq T_A\leq T_{MAX}$	**+10.0**		**+9.8, +10.2**	mV/˚C
Load Regulation	T_A=+25˚C	±0.4	±2.0		mV/mA
$\leq I_L\leq$1 mA T	$_{MIN}\leq T_A\leq T_{MAX}$	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25˚C	±0.01	±0.1		mV/Va
	$\leq V_S\leq$30V	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25˚C	56	80		µA
	$_S$=+5V	**91**		**138**	µA
	V_S=+30V, +25˚C	56.2	82		µA
	V_S=+30V	**91.5**		**141**	µA
Change of Quiescent Current	4V≤$V_S\leq$30V, +25˚C	0.2	2.0		µA
	4V≤$V_S\leq$30V	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	˚C
Long Term Stability	$T_J=T_{MAX}$, for 1000 hours	±0.08			˚C

Tabulka 3 Charakteristiky teploměru LM35DZ.(National Semiconductor, 2000)



Obrázek 5.3 Přechodová charakteristika teploměru v prostředí stálého vzduchu. (National Semiconductor, 2000)

Teploměry jsou spojeny s měřicí kartou LabJack UE9 pomocí kroucené dvojlinky. Protože zvolené teploměry nejsou náročné zařízení na přívodní napájení, je možné je napájet pomocí měřicí karty. Vstupní napětí je nastaveno na 5V. Měřicí karta je spojena se serverem pomocí ethernetu.

5.3 Řešení na straně serveru

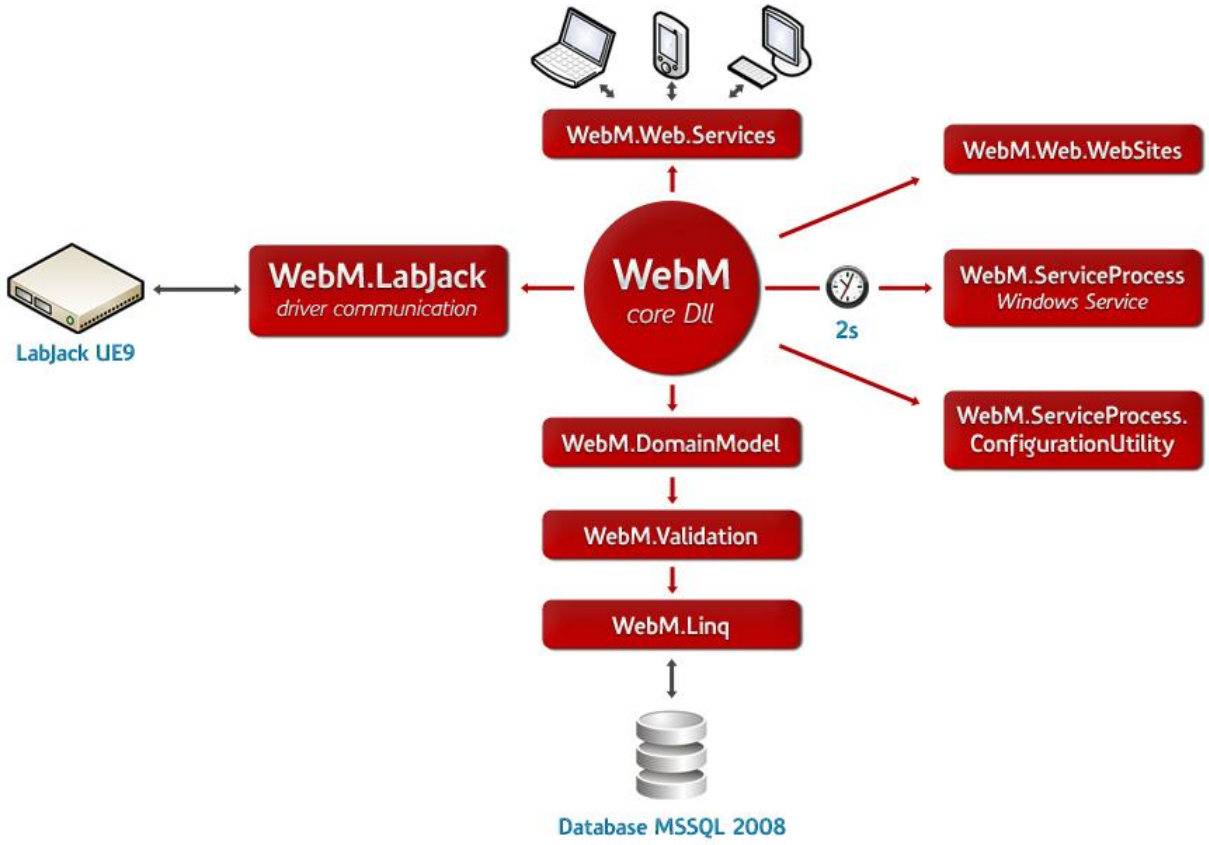
Tato část systému pro měření teploty má za úkol hned několik důležitých úkolů jako například:

- komunikaci s měřicí kartou LabJack UE9,
- zpracování naměřených hodnot,
- převod měřeného napětí na stupně Celsia,
- ukládání hodnot do SQL databáze,
- poskytování dat pro klienty atd.

Jako databázový server jsem zvolil Microsoft SQL Server 2008, na kterém je umístěna jedna databáze. Databázový server je nakonfigurován pro vzdálené připojení.

Server musí zpracovávat požadavky klientů přicházející ze sítě Internet. Zpracovávání požadavků je řízeno serverem Microsoft IIS (Internet Information Server). Ten bylo zapotřebí nakonfigurovat a zabezpečit.

Celá aplikace byla vytvořena jako jedna solution postavená na Microsoft .Net Framework verze 3.5. Jako společný programovací jazyk pro všechny aplikace jsem zvolil jazyk C# a solution byla vytvořena pomocí Microsoft Visual Studia 2008. Aplikace je umístěna v namespace s názvem WebM a jednotlivé aplikace sdílejí mezi sebou své součásti a třídy. Schéma jednotlivých částí ukazuje následující diagram:



Obrázek 5.4 Architektura jmenného prostoru WebM

Namespace WebM

V tomto nejrozsáhlejším namespace se nachází třídy pro práci s měřicí kartou LabJack UE9, třídy pro komunikaci s databází a další podpůrné třídy.

Mezi zajímavá řešení patří například použitý Entity Framework. Aplikace je postavena na Entity Frameworku používající technologii LINQ to SQL. Mé řešení

nepoužívá třídy generované pomocí Visual Studia, ale používá své vlastní řešení Entity Frameworku. Každá z entit je poděděna z abstraktní třídy *BaseEntity* nacházející se v namespace *WebM.DomainModel*. Zkrácený příklad kódu entity může vypadat následovně:

```
using WebM.Validation;
...
namespace WebM.DomainModel
{
    [Table(Name = "WebM.Values")]
    public class Value : BaseEntity
    {
        #region Properties

        [Column(AutoSync = AutoSync.OnInsert, IsPrimaryKey = true, IsDbGenerated = true)]
        [XmlIgnore]
        public int ValueId { get; set; }

        private double _Temperature;
        [Column(Storage = "_Temperature")]
        public double Temperature
        {
            get { return _Temperature; }
            set
            {
                if (_Temperature != value)
                {
                    _Temperature = value;
                    PropertyHasChanged();
                }
            }
        }

        [Column]
        [XmlIgnore]
        public int SensorId { get; private set; }

             ...

        #endregion

        #region Constructors

        public Value(double temperature, Sensor sensor)
        {
            _Temperature = temperature;
            SensorId = (int)sensor;
        }

        protected override void AddBusinessRules()
        {
            ValidationRules.AddRule(
                      CommonRules.MinValue<double>,
                      new CommonRules.MinValueRuleArgs<double>("Temperature", 0));
            ValidationRules.AddRule(CommonRules.MaxValue<double>,
                       new CommonRules.MaxValueRuleArgs<double>("Temperature", 100));
        }

       ...

        #endregion
```

```
        #region Methods

          public bool Save()
          {
              ValidationRules.CheckRules();
              if (IsSavable)
              {
                  if (IsNew)
                      Context.Values.InsertOnSubmit(this);

                  Context.SubmitChanges();
                  MarkOld();
              }
              return IsValid;
          }

          public static List<Value> GetValues(int count, Sensor sensor)
          {
              return Context.Values
                .Where(value => value.SensorId == (int)sensor)
                .OrderByDescending(value => value.CreationDate)
                .Take(count)
                .ToList();
          }

          #endregion
      }
}
```

Takto vytvořená entita v sobě zahrnuje jak statické třídy pro získávání hodnot, tak i metody pro ukládání do databáze a serverovou validaci. Validace je řešena pomocí tříd v namespace WebM.Validation a využívá metody reflection. Práce s takto vytvořenou třídou je pak velice snadná. Následující zdrojový kód na pouhé dva řádky získá z měřicí karty hodnotu napětí na třetím senzoru, hodnotu zpracuje a převede na [°C], validuje, a pokud je vše v pořádku, hodnoty se uloží do databáze.

```
Value value = new Value(_LabJackUE9.GetAnalogValue(2), 2);
value.Save();
```

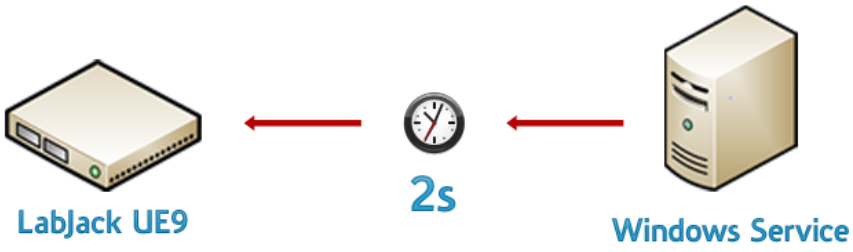
Ukázka kompletního kódu pro jednu z metod webové službu pro získání hodnot je následující:

```
[WebMethod]
public List<Value> GetTemperatures(int count, Sensor sensor)
{
  return Value.GetValues(count, sensor);
}
```

Jak je vidět na zdrojovém kódu výše, je celý systém pomocí navržené struktury velice snadný a jednoduchý. Tímto se stává celá aplikace velice snadno upravitelnou a rozšiřitelnou a čitelnou.

Namespace WebM.ServiceProcess

V tomto jmenném prostoru se nachází služba systému Windows pro kontinuální měření teplot. Úkol této služby je ten, že každé 2 sekundy se dotáže měřicí karty na aktuální hodnoty napětí na teploměrech a hodnoty uloží do databáze.

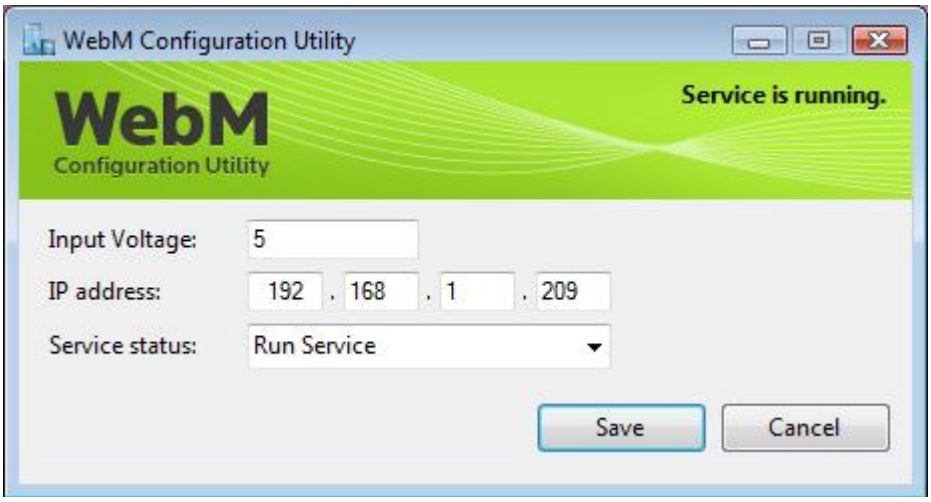


Obrázek 5.5 Služba systému Windows komunikuje s měřicí kartou LabJack každé 2s

Služba je pojmenována *WebM*. Protože k funkcím měřicí karty nepřistupuje jen tato služba, ale i webová služba je zapotřebí nastavení pro měření ukládat na vhodné místo. Jako jednu z více možných variant jsem zvolil registry systému Windows. V těchto registrech je uložené napájecí napětí pro teploměry a cílová IP adresa měřicí karty. Protože správa webových služeb a registrů systému Windows je docela komplikovaná, vytvořil jsem pro tyto účely pomocnou konfigurační aplikaci s názvem WebM Configuration Utility.

WebM Configuration Utility

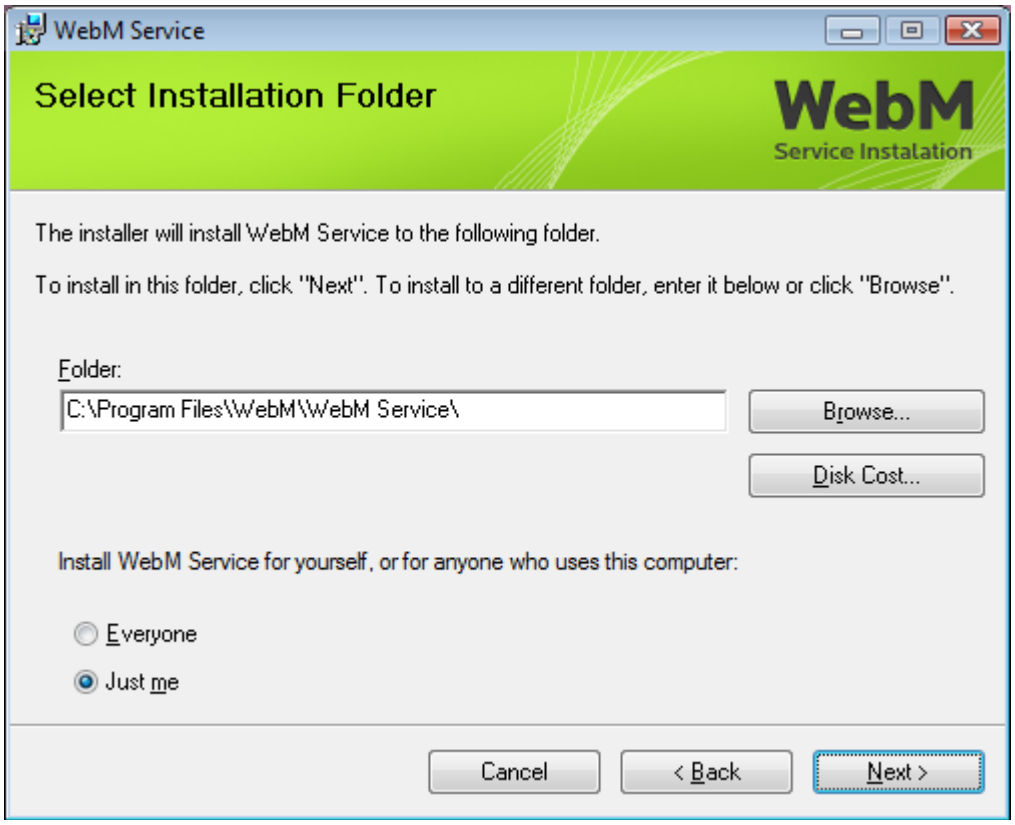
Tato pomocná aplikace usnadňuje ovládání měřicí služby a editaci potřebných klíčů v registrech systému Windows.



Obrázek 5.6 Uživatelské rozhranní pomocné konfigurační aplikace

Instalační balíček

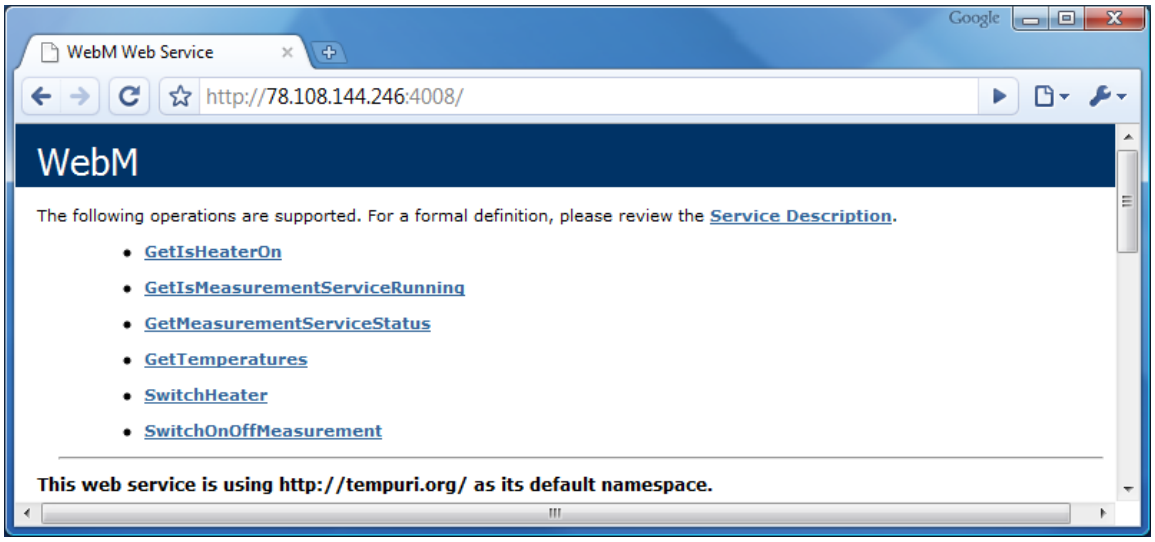
Protože instalace služby do systému není zcela jednoduchá, vytvořil jsem pomocnou aplikaci, která vše zjednodušuje. Během instalace se vytvoří potřebné klíče v registru Windows a spolu se službou pro měření se nainstaluje i WebM Configuration Utility.



Obrázek 5.7 Proces instalace měřicí služby a konfigurační aplikace

WebM.Web.Services

V tomto namespace se nachází webová služba poskytující data pro klienty. Pomocí několika metod, umožňuje také získat informaci, zda měřicí služba je spuštěna.



Obrázek 5.8 Výchozí stránka webové služby s odkazem na WSDL dokument

5.4 Klientská aplikace pro mobilní zařízení

Jako ukázku klienta konzumující webovou službu jsem připravil aplikaci pro mobilní zařízení. Aplikace je postavena na technologii Adobe Flash Lite 2.0. Tuto technologii jsem zvolil především pro její univerzálnost.

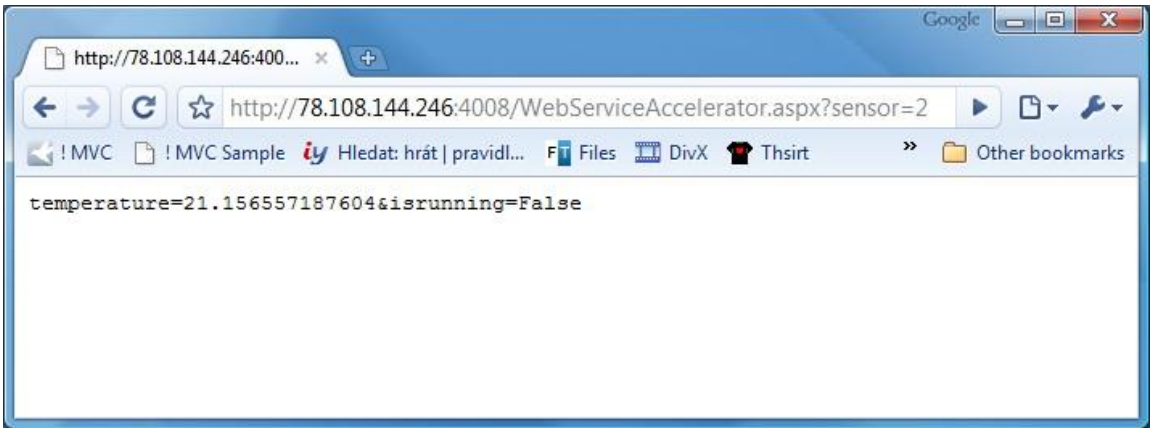
Nová komponenta je kompletně přepsána, jsou použity nové vylepšené algoritmy pro vykreslování, má menší velikost a několika násobně větší rychlost zpracování. Aplikace umožňuje běh ve dvou módech *XML Web Service* a tzv. *rychlý mód*.

V prvním jmenovaném módu zpracovává mobilní klient data jako XML zprávy z webové služby. Zkrácený kód programovacího jazyka ActionScript 2.0 pro volání XML webové služby a vykreslení grafu je následující:

```
ServiceCall = WebMService.GetTemperatures(1, SensorId);
ServiceCall.onResult = function(result) {
        var dataValue:Number = int(Number(result.xmlNodes[0].firstChild.firstChild.nodeValue)*10)/10;
      if (Values.length == 0) {
            for (var i:Number = 0; i < DataCount; i++) {
                  Values.push(0);
            }
      }

      if (IsServiceRunning) {
            Values.shift();
            Values.push(dataValue.toString() == 'NaN' ? 0 : dataValue);
      }

      MaxValue = getMaxValue();
      DrawInit();
      DrawChart();
      DrawSummary();
      DrawServiceStatus();
};

ServiceCall.onFault = function(fault:SOAPFault) {
      trace(fault.faultstring);
      trace(fault.detail);
};
```

Nejzajímavějším řádkem kódu je hned první. ActionScript volá přímo metodu *GetTemperatures*, se vstupními parametry pro počet požadovaných hodnot a číslo senzoru. O posílání SOAP zprávy s požadavkem na návratové hodnoty se stará přímo ActionScript resp. Flash.

Druhý rychlý mód přistupuje k datům pomocí speciálně vytvořené aplikaci nazvané *WebServiceAccelerator* vytvořené pro snížení datového toku mezi serverem a mobilním zařízením. WebServiceAccelerator je umístěn v namespace WebM.Web.Services. WebServiceAccelerator pracuje tak, že XML webovou službu zpracuje a vrátí textový dokument ve formě prostého textu.



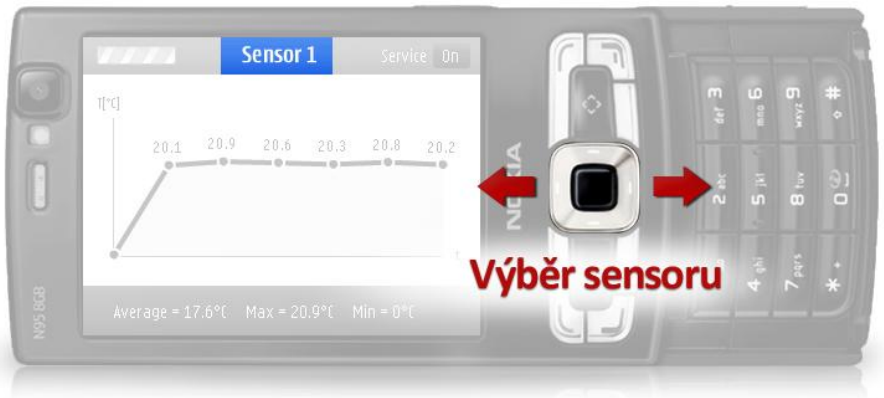
Obrázek 5.9 Výstup WebServiceAccelerator ve formě prostého textu.

Za pomocí tohoto akcelerátoru se odstraní nevýhoda velkého datového toku SOAP zpráv a výrazně se sníží datový tok. Nevýhodou je ztráta robustnosti. WebServiceAccelerator je přímo určený pro mobilního klienta.

Snížením datového toku se zvýší i rychlost mobilního klienta. Při podmínkách dobrého signálu GPRS dokáže aplikace dosáhnout odezvy až 0.3s. Protože se jedná o tepelný proces, kde použitý teploměr vyžaduje přibližně 8 sekund pro ustálení hodnoty (viz. Obrázek 5.3 Přechodová charakteristika teploměru v prostředí stálého vzduchu), jako dostatečnou obnovovací periodu jsem zvolil 4s.

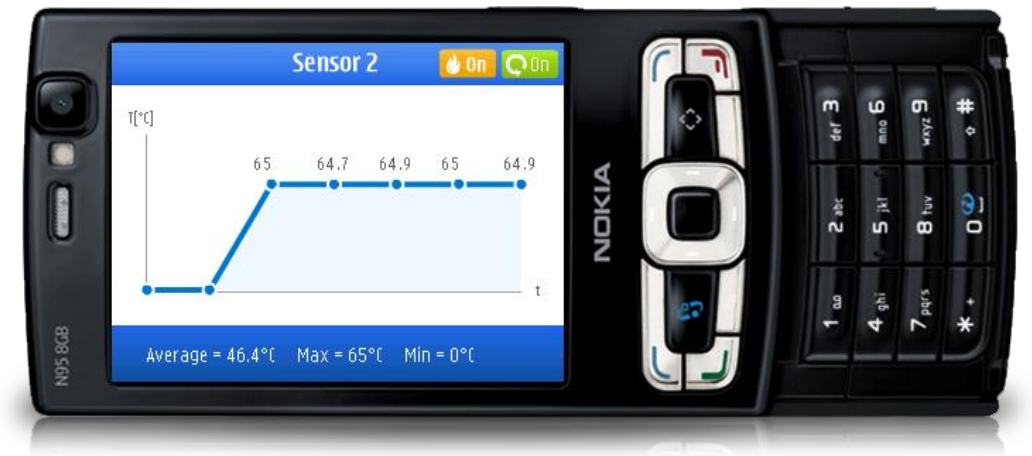
5.5 Popis softwaru pro mobilní telefony

Ovládání funguje pomocí kláves mobilního telefonu. Šipkami *doleva* a *doprava* program přepíná zvolený senzor, klávesami 1 až 4 vybírá senzor přímo. Uživatelské rozhraní softwaru pro mobilní zařízení je ukázáno na následujícím obrázku.



Obrázek 5.10 Mobilní klient spuštěný na telefonu Nokia N95.

Obrázek 5.11 Mobilní klient spuštěný na telefonu Nokia E71.



Obrázek 5.12 Mobilní klient spuštěný na telefonu Nokia N95.

Závěr

Tato práce se zabývá možnostmi XML webových služeb v prostředí automatizace a jejich tvorbou.

První část je věnovaná popisu technologie webových služeb pro sdílení obchodní logiky. V této kapitole je popsán historický vývoj této univerzální technologie, formy přenosu dat mezi aplikacemi, její standardizovaný dokument pro přenos zpráv SOAP i její standardizovaný dokument pro popis služby WSDL, jejich funkcí a vstupních i výstupních parametrů. K tomuto dokumentu, který se zkratkou nazývá WSDL je připojena i zkrácená ukázka.

Další část mé práce se zabývá analýzou komunikace s ohledem na prostředí programovacího jazyka C# resp. prostředí Microsoft .Net Framework. Věnuji se zde možnostem sériové komunikace po standardu RS 232, standardu USB i komunikaci pomocí protokolu TCP.

V kapitole 3 je popsána měřicí karta LabJack UE9, která je použita pro následující dvě praktické ukázky použití webových služeb.

První z nich je měření úhlu natočení pomocí odporového snímače. Tato aplikace využívá zařízení společnosti LabJack UE9 připojené k serveru komunikující TCP/IP protokolem. K této úloze bylo nutné naprogramovat službu operačního systému Windows ke sběru dat z tohoto zařízení, webovou službu, která zprostředkovává tato naměřená data a přistupuje i k tomuto zařízení a zprostředkovává tak funkci například pro ovládání světla. K této webové službě jsem vytvořil i dva klienty. První přistupuje k webové službě jako webová stránka a druhý je pro nasazení v mobilních telefonech.

Druhou praktickou ukázkou je systém pro měření teploty. Úloha je rozdělena do tří hlavních částí.

V první z nich pojmenovaná jako hardwarová vrstva bylo mým úkolem zvolit a sestrojit 4 teploměry. Výsledné zapojení teploměrů LM35DZ je vhodné pro měření na větší vzdálenosti a doba ustálení teploměru je přibližně 8s.

Další vrstvou tohoto systému je řešení na straně serveru. Pro tento účel jsem naprogramoval v jazyce C# ucelené řešení s názvem WebM, které sdílejí mezi sebou

všechny ostatní části systému. Toto řešení je pokročilejší a velmi vylepšuje serverovou část oproti systému měření úhlu natočení. Výsledkem komponenty WebM je efektivní práce v kódu a jednoduchá rozšiřitelnost systému.

Poslední částí druhé praktické je tvorba klienta pro mobilní zařízení. Opět je oproti původní komponentě značně vylepšen. Výsledkem je funkční aplikace pro mobilní telefony, která vykresluje aktuální průběh naměřených dat a umožňuje přepínat mezi 4 snímači tepla.

Použitá literatura

ADOBE SYSTEMS. 2007. *Getting started with Flash Lite 2.x*. [Online] 2007. [Citace: 4. 11 2008.] http://livedocs.adobe.com/flash/9.0/main/flashlite2_gettingstarted.pdf.

ADOBE SYSTEMS. 2007. *Supported Devices. Adobe*. [Online] [Citace: 20. 12 2008.] http://www.adobe.com/mobile/supported_devices/.

BEALE, JOHN. 2007. *USB Power for RH1 HiMD from external batteries*. [Online] 2007. [Citace: 15. 2. 2008] http://www.bealecorner.org/best/measure/USB/index.html.

COHEN, ELI. 2005. *Issues in Informing Science and Information Technology*. 2005. str. 786. ISBN 83-922337-0-0.

DANTRE, TAPAN. 2004. The *Code Project. Serial Communication using C# and Whidbey*. [Online] 2004. [Citace: 06. 5. 2007] http://www.codeproject.com/csharp/SerialCommunication.asp.

DUTHIE, G. A. 2003. *ASP.NET Krok za krokem*. Praha : Mobil Media, 2003. str. 512. ISBN 80-86593-33-9.

HW SERVER. 2003. *HW server představuje - RS-232*. [Online] 2003. [Citace: 6. 5. 2007] http://rs232.hw.cz/.

HW SERVER. 2005. *USB/Ethernet měřicí karta LabJack UE9*. [Online] 2005. [Citace: 4. 3. 2008] http://hw.cz/Produkty/Obecne-produkty/ART1349-USB-Ethernet-merici-karta-LabJack-UE9.html.

HYDE, JOHN. *Learning USB by Doing*. [Online] 2006. [Citace: 4. 4. 2008] http://www.devasys.com/PD11x/JHWP.pdf.

KAČMÁŘ, DALIBOR. 2001. *Programujeme .NET aplikace ve Visual Studiu .NET*. Praha : Computer Press, 2001. ISBN 80-7226-569-5.

KRCS. *USB Cable - Type A to B*. [Online] [Citace: 15. 2. 2008] https://www.krcs.co.uk/images/library/products/300/lindy_31646.jpg.

LABJACK CORPORATION. *LabJack UE9*. [Online] [Citace: 4. 11. 2008] http://www.labjack.com/labjack_ue9.php?prodId=56.

MACDONALD, MATTHEW A SZPUSZTA, MARIO. 2006. *ASP.NET 2.0 a C# - tvorba dynamických stránek profesionálně*. [překl.] RNDr. Jan Pokorný, Ing. Petr Kadlec a Erika Lencová. Praha : Zoner Press, 2006. ISBN 80-86815-38-2.

MANES, ANNE THOMAS. 2003. *Web services: A Manager's Guide*. 2003. ISBN 0-321-185377-3.

MICROSOFT CORPORATION. 2005. *Obecné informace o protokolu TCP/IP*. [Online] 2005. [Citace: 2. 4. 2009] http://www.microsoft.com/technet/prodtechnol/windowsserver2003/cs/library/ServerHelp/43f255c5-e33a-46cf-b4bc-925a5599640a.mspx?mfr=true.

MICROSOFT CORPORATION. 2005. Protokol *TCP (Transmission Control Protocol)*. [Online] 2005. [Citace: 2. 4. 2009] http://www.microsoft.com/technet/prodtechnol/windowsserver2003/cs/library/ServerHelp/43f255c5-e33a-46cf-b4bc-925a5599640a.mspx?mfr=true.

MICROSOFT CORPORATION. *System.IO.Ports Namespace*. [Online] [Citace: 6. 5. 2007] http://msdn2.microsoft.com/en-us/library/system.io.ports.aspx.

MICROSOFT CORPORATION. *System.Net.Sockets Namespace*. [Online] [Citace: 10. 5. 2007] http://msdn2.microsoft.com/en-us/library/system.net.sockets.aspx.

MICROSOFT CORPORATION. *Windows Image Acquisition (WIA)*. [Online] [Citace: 6. 3. 2007] Microsoft Corporation. http://msdn.microsoft.com/en-us/library/ms630368(VS.85).aspx

NATIONAL SEMICONDUCTOR. 2000. *Precision Centigrade Temperature Sensors*. [Online] 2000. [Citace: 7. 4. 2008] http://www.gme.cz/_dokumentace/dokumenty/313/313-909/dsh.313-909.1.pdf.

O'RIORDAN, DAVID. 2002. *Business Process Standards for Web Services*. WebServicesArchitect.com. [Online] 2002. [Citace: 5. 10. 2008] http://www.webservicesarchitect.com/content/articles/oriordan01.asp.

PEACOCK, CRAIG. 2002. *USB in a Nutshell - Making sense of the USB standard*. [Online] 2002. [Citace: 14. 3. 2008] http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf.

PROSISE, J. 2003. *Programování v Microsoft..NET*. Praha : Computer Press, 2003. str. 736. ISBN 80-7226-879.

RANKL, WOLFGANG. 2004. *SMS Transceiver for PIC*. [Online] 13. 10. 2004. [Citace: 14. 3 2009.] http://www.wrankl.de/SMST4PIC/SMST4PIC.html.

ŘEHÁK, JAN. 2002. *USB - Universal Serial Bus - Popis rozhraní*. HW.cz. [Online] 7. 5. 2002. [Citace: 14. 2 2008.] http://hw.cz/Teorie-a-praxe/Dokumentace/ART327-USB---Universal-Serial-Bus---Popis-rozhrani.html.

SCHEIDEGGE, THOMAS. *WIA Scripting and .NET. The Code Project*. [Online] [Citace: 10. 5. 2007] http://www.codeproject.com/dotnet/wiascriptingdotnet.asp.

SHREVE, JUSTIN G. *Základy webových služeb*. [Online] 2005. [Citace: 7. 4. 2007] https://developer.mozilla.org/cs/Základy_webových_služeb.

TIŠNOVSKÝ, PAVEL. *Sériový port RS-232C*. [Online] 27. 11. 2008. [Citace: 12. 3 2009.] http://www.root.cz/clanky/seriovy-port-rs-232c/.

USB IMPLEMENTERS FORUM, INC. *Universal Serial Bus Specification*. [Online] 27. 4 2000. [Citace: 2. 4 2009.] http://www.usb.org/developers/docs/usb_20_042409.zip.

VISHAY SPECTROL. *7/8" (22mm) Precision Wirewound Potentiometer*. [Online] 9. 8. 2004. [Citace: 4. 4. 2008] http://www.gme.cz/_dokumentace/dokumenty/113/113-052/dsh.113-052.1.pdf.

W3C. *Web Services Architecture. W3C Working Group Note 11 February 2004*. [Online] 2004. [Citace: 9. 4. 2007] http://www.w3.org/TR/ws-arch/.

WIKIPEDIA. *SOAP*. [Online] 7. 3. 2007 [Citace: 7. 3. 2007] http://en.wikipedia.org/wiki/Simple_Object_Access_Protocol.